A METHOD FOR PRECISE ROBOT CYCLE TIME ESTIMATION

by

LAURA L. CRANMER

B.S., Kansas State University, 1988

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering
KANSAS STATE UNIVERSITY
Manhattan, Kansas
1989

Approved by:

Doris Grant

Major Professor

A11208 617585

ACKNOWLEDGEMENT

I wish to acknowledge the invaluable assistance of Professor Brad Kramer at Kansas State University, Dr. Jack Henry of IBM-Austin and the support from the IBM-KSU Research Project team members. I also wish to thank my committee members: Professor Doris Grosh, Professor L.E. Grosh and Professor Mark McNulty.

TABLE OF CONTENTS

ACKNOWLEDGEMENT i

LIST OF FIGURES v

LIST OF TABLES vi

CHAPTER 1: INTRODUCTION

1.0 IBM-KSU Research Project Description 1

1.1 Contents 2

Pick and Place Application 4

1.2 Organization 5

CHAPTER 2: LITERATURE SEARCH

2.0 Importance of This Work 7

Installation of New Systems 7

Changeovers 8

2.1 Cycle Time Estimation Procedures. 9

Tanner's Rule of Thumb 10

ROFAC - RTM Method 11

Ahmadi - CNC Pick and Place Machine . . . 11

Steiner - PUMA 500 12

SIM/7535 13

2.2 Summary 13

CHAPTER 3: ESTIMATING IBM 7547 SCARA ROBOT CYCLE TIME

3.0 IBM 7547 SCARA Robot Description. 15

3.1 Step 1: Exploring the 7547 SCARA Joint Movement 17

Test Methodology - distance tests 17

3.2 The Control Program Hypotheses 18

Hypothesis 1 18
Hypothesis 2 19
Hypothesis 3 19
3.3 Step 2: Estimating Joint Speeds 20
Joint Tests 21
3.4 Step 3: Regression Analysis 22
3.5 Step 4: Develop Cycle Time Estimate Algorithm . 24
Logic for Time Estimate of Simple Point to Point Move 25
3.6 End of Arm Movement: Z Axis 29
3.7 Step 5: The Emulator Program. 30
3.8 Step 6: Verify Emulator 35
3.9 Summary 38
CHAPTER 4: A GENERAL METHOD FOR ESTIMATING ROBOT CYCLE TIME
4.0 The Method. 40
Step 1 40
Step 2 41
Step 3 43
Step 4 43
Step 5 44
Step 6 44
4.1 Summary 44
CHAPTER 5: CONCLUSION AND FUTURE CONSIDERATIONS
5.0 Summary 46
5.1 Future Applications 47
5.2 Future Research 48

REFERENCES 49

APPENDIX

IBM Proposal 53

Distance Test Data 60

Joint Tests 64

Joint Test Average Speeds 66

Regression Equations 68

Z-axis Tests 77

Z-axis tests for 7547 Application 78

Magazine Status 79

Regression Equations Verified - Additional Data 81

AML Program 84

Payload Data 98

ABSTRACT TITLE PAGE 101

ABSTRACT 102

LIST OF FIGURES

Figure 1.1:	Process Mock-up	4
Figure 3.1:	IBM 7547 SCARA Robot	16
Figure 3.2:	Work Station Test Locations	17
Figure 3.3:	IBM 7547 SCARA Robot Tool Tip Speeds	20
Figure 3.4:	Theta 1 Joint Tests	21
Figure 3.5:	Theta 2 Joint Tests	21
Figure 3.6:	Work Envelope	25
Figure 3.7:	Time Estimate Logic	26
Figure 3.8:	Time Estimate Logic	27
Figure 3.9:	Angular Movement	28
Figure 3.10:	Work Station Layout	31
Figure 3.11:	Robot Moves	32
Figure 3.12:	BOM Appended Version	32
Figure 3.13:	Emulator Flowchart	34
Figure 3.14:	Modified Emulator & AML Program	35
Figure 3.15:	Time for Various Payloads	37
Figure 3.16:	Emulator Tests	38
Figure 4.1:	Maximum Tool Tip Speed Not Reached	42
Figure 4.2:	Maximum Tool Tip Speed Reached	42

LIST OF TABLES

Table A1:	30 mm Time-Distance Data	60
Table A2:	100 mm Time-Distance Data	61
Table A3:	250 mm Time-Distance Data	62
Table A4:	500 mm Time-Distance Data	63
Table A5:	Theta 1 Joint Tests	64
Table A6:	Theta 2 Joint Tests	65
Table A7:	Theta 1 Average Speed	66
Table A8:	Theta 2 Average Speed	67
Table A10:	Regression Equations	68
Table A11:	Z-axis Tests	77
Table A12:	Z-axis Tests for 7547 Application . .	78
Table A13:	Magazine Status Verified	79
Table A14:	Theta 1 Regression Equations Verified	81
Table A15:	Theta 2 Regression Equations Verified	83
Table A16:	Theta 1 Payload Data	98
Table A17:	Theta 2 Payload Data	99
Table A18:	Z-axis Payload Data	100

CHAPTER 1: INTRODUCTION

This investigation into robot cycle time estimation has been the result of joint sponsorship effort by IBM and KSU.

1.0 IBM-KSU RESEARCH PROJECT DESCRIPTION

In March 1988 a contract was signed between Kansas State University (KSU) and IBM. IBM proposed the "development of a set of models and programs for optimized planning and operation of the IBM 7547 SCARA Robot selected for 'plate loading' operation automation [1]."

The plate loading operation is a pick and place operation. The task involves the removal of empty dispensers from a fixture plate followed by the replacement with full dispensers. Each dispenser is placed and held at a precise location on the plate with a given orientation. All dispensers are identical in shape and size. Dispensers, however, are part-type specific [1]. This operation is automated due to the large number of dispensers per plate and the dispenser's small size and identical appearance.

The plate loading operation is a bottleneck and therefore was chosen for optimization. IBM provided a project description and two industrial robots to complete this project. (The project description is included in the Appendix.) For project completion IBM required a set of

algorithms to:

1) Determine an optimal path for the robot to follow
2) Determine the optimal robot layout
3) Accurately estimate the robot cycle time

Optimal path and layout algorithms are discussed in both the Final Report to IBM [7] and Larry Satzler's KSU Master's Thesis [22]. The method developed to precisely estimate the robot cycle time is the subject of this thesis.

The IBM-KSU research project required the development of a robot emulator to verify the cycle time estimation method [1]. The robot emulator is a software characterization of the robot's processes. The emulator's purpose is to provide an accurate cycle time prediction for a given layout and pick and place sequence. The emulator must provide a cycle time prediction within $\pm$5% of the actual cycle time. This precision is vital for the emulator to be of value as a planning tool.

1.1 CONTENTS

A method to develop precise robot cycle time estimates for an unknown device was needed to complete the IBM-KSU research project. This method had to be quick and simple to meet the contract time requirements. A literature search was conducted and an accurate method was not found. Current literature confirmed the importance of estimating robot cycle

time. Robot cycle time estimates are a valuable planning tool. These estimates are used to determine system capacity, balance production, and establish schedules. Precise estimates are critical in highly integrated automated systems as process rates depend heavily on robot cycle time [8,17].

Several articles presented cycle time estimation methods based on known control systems [12,17,18]. Simulation packages are also available that estimate cycle time; however, these do not provide the necessary accuracy [13,20]. This inaccuracy may be due to variations in the manufacturing process, end-of-arm rate, mass of payload, and/or an inaccurate characterization of the robot control system [20,13]. Additionally, precision modeling of robot arms requires information on the characteristics of the particular control system which manufacturers are unwilling to release [13].

This thesis introduces a quick and simple method to precisely estimate robot cycle time without knowledge of the control system. The robot is essentially treated as a black box. The cycle time estimate is based on the development of equations which describe the average speed reached by each joint for a specific move. This method was applied to an IBM 7547 SCARA Robot using a pick and place assembly application.

Pick and Place Assembly Application

The robot's function in this application is to replenish a fixture plate with new dispensers as quickly as possible. The robot removes an empty dispenser from the plate, places the empty dispenser in the empty position in a cartridge, retrieves a full dispenser from that cartridge, and places the full dispenser on the plate. This process continues until all empty dispensers have been replaced with full dispensers. (See Figure 1.1.)

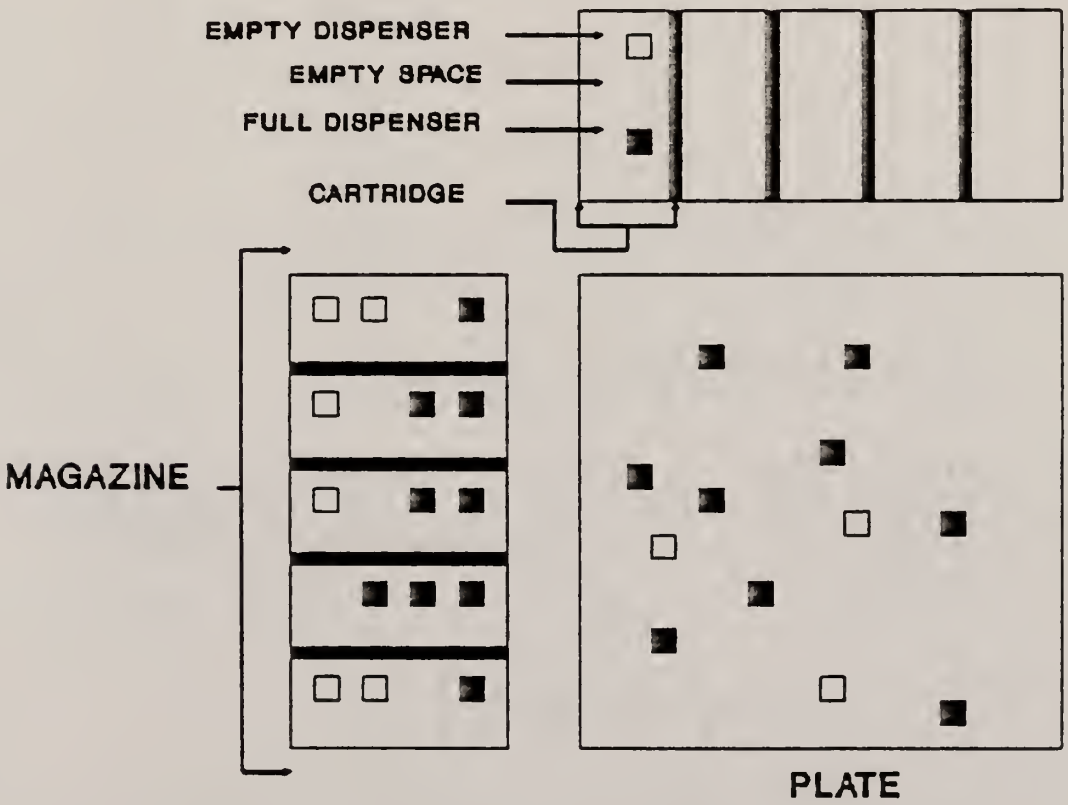


Figure 1.1: Process Mock-up

The dispensers hold components used in the assembly of printed circuit boards. Each dispenser holds one type of component. A cartridge stores one dispenser type, and a set of cartridges forms a magazine. The fixture plate holds the dispensers for the circuit board assembly process. The operator manually loads and unloads the plates and cartridges.

An IBM 7547 emulator was developed to verify the cycle estimation method in the pick and place assembly application. The emulator uses the current layout, error handling routines, and a bill of materials (BOM) for a given plate to develop the pick and place sequence. The emulator uses this sequence to estimate the cycle time. The IBM 7547 emulator provides an accurate cycle time estimate (within $\pm$1%) in considerably less time than required for the robot to perform the task. The 7547 emulator is discussed in detail in Chapter 3.

The emulator was also used in the IBM-KSU project to evaluate various pick and place heuristics. This allowed the evaluation of hypotheses without having to operate the 7547 SCARA Robot and saved a great deal of time.

1.2 ORGANIZATION

This report is organized as follows: Chapter 2 discusses the value of cycle time estimation and presents a survey of available cycle time estimation methods. Chapter 3

presents the process applied to estimate cycle time for an IBM 7547 SCARA Robot. Chapter 4 presents a general method to precisely estimate robot cycle time. Chapter 5 summarizes the completed work and recommends future applications and research.

CHAPTER 2: LITERATURE SEARCH

A quick and simple method to estimate precisely robot cycle time was required to complete the IBM-KSU research project. This method had to be empirically based since the control system characteristics were unknown. A literature search for an established method was conducted. Numerous articles were found describing the value of accurately estimating robot cycle time, but, no information was available detailing a precise cycle time estimation method for an unknown device.

2.0 IMPORTANCE OF THIS WORK

The manufacturing sector emphasizes operations flexibility. The goal is a quick response to changes in product mix and market demands. Precise robot cycle time estimates are a tool to minimize response time and increase flexibility [13,21]. These estimates minimize response time by allowing the engineer to determine quickly system capacity, balance production and establish schedules. Cycle time estimates are valuable when installing new systems and initiating a task changeover.

Installation of New Systems

Robot application planning is extremely complex due to the variety of subproblems and the wide range of

technological and economic knowledge required [26]. A great deal of time, money and effort is spent when installing robot automation. Approximately sixty to eighty percent of the total effort is involved in layout, equipment design, robot selection and hardware mock-up [20].

Robot cycle time estimates are useful when installing a new system. Accurate cycle time estimates allow the engineer to evaluate various robots, test the layout, and test the program (path choice) all prior to installation [20,14,16]. Cycle time estimates are also useful to determine system capacity, balance production, manage inventories and establish schedules [21].

Changeovers

Accurate cycle time estimates also minimize task changeover times. Manufacturers must be able to modify quickly the production line to respond to a dynamic market. Cycle time estimates can be used to determine quickly the best method to perform the new task [20,2,3]. Precise estimates are critical since operations time and volume depend heavily on the cycle time of the robot [5]. Accurate cycle time estimates are used by IBM for this application to [2,3]:

1. Determine process rates
2. Pre-evaluate proposed engineering changes

3. Construct setups and sequences

4. Develop parameters for line-balancing procedures

2.1 CYCLE TIME ESTIMATION PROCEDURES

Several articles describe empirical methods to estimate robot cycle time. These methods provide estimates with varying accuracy levels. Simulation packages are also available which estimate cycle times. Most do not specify the method or accuracy of the cycle time estimates [4,9,10,16,23,26].

Peck investigated a number of simulation packages (Intergraph, MCL, ROBOCAM, and PLACE) and found these packages do not estimate cycle times accurately [20]. Peck suggested a variety of reasons for the inaccuracy. The same model of robots manufactured by a given vendor can vary due to manufacturing practices. Most simulation packages don't control the end-of-arm rate with any accuracy. Most packages also ignore the mass of the payload. In general, simulation packages do not accurately characterize the controller.

Iversen notes that the problem with available simulation packages has been how to model robot arms accurately without knowledge of the detailed characteristics of the robot control system [13]. Manufacturers will not disclose the necessary control information since this information gives the manufacturer a proprietary edge. Most manufacturers do provide machine specifications. These include idealized

process rates and maximum joint velocities. These specifications, however, do not provide enough information on which to base an adequate systems analysis [2,3]. Thus the problem remains how to experimentally characterize the robot and develop precise cycle time estimates.

Several authors have developed cycle time estimation methods based on discrete time estimates for a particular move. Many authors, however, did not include a comparison of actual time versus estimated time. Tanner's Rule of Thumb, the RTM method applied in ROFAC, and Ahmadi's CNC emulator are three examples which are described below.

Tanner's Rule of Thumb

W.R. Tanner introduced the following rule of thumb for estimating robot cycle time [11]:

1 second per move

1 second per major change of part orientation

0.5 second at end of the path for positioning

0.5 second for each use of the gripper

Tanner's heuristic does not consider robot capabilities, movement length, or speed and thus provides a very rough cycle time estimate. This method would be useful only when accurate estimates are not required.

ROFAC--RTM Method

Hershey et al. present ROFAC (Robot Factor) a software package [11]. The time estimation portion of ROFAC (ROFAC/TIME) is based on the RTM (Robot Time and Motion) method. RTM is analagous to MTM (methods time measurement) where discrete time elements are assigned to each move. Time element tables are developed based on experiments with a specific robot [19]. These tables are incorporated into the ROFAC/TIME software. ROFAC/TIME was specifically developed for the Cincinnati Milacron T3 and the Unimation PUMA robots. ROFAC/TIME was tested on seven industrial applications, and the cycle time predictions were accurate within $\pm$8.6%.

Ahmadi--CNC Pick and Place Machine

Ahmadi et al. presented an emulator which estimates cycle time of a CNC pick and place machine [2,3]. The emulator was required to complete performance optimization procedures in a surface mount application. Many of the CNC operations are completed concurrently and other operations are dependent upon the completion of previous operations. This makes modeling difficult.

The critical process parameters and their relationships were first identified. Discrete time estimates for the operation characteristics of the machine were developed using a set of optically triggered counters [2]. These counters

provided estimates within ±0.02 seconds. Five potential scenarios are defined, each with a different control logic. The order and manner by which a component is picked up and placed, along with the arm, board and carrier movements determine which scenerio is used.

Finally, an emulator was developed to verify the system characterization. The emulator used the current system layout and pick and place sequence. This method was shown accurate within 95% to 99%.

Steiner--PUMA 500

Another approach to cycle time estimation was presented by Steiner of General Electric [24]. The velocity and acceleration of each joint were experimentally determined using strip chart recorders. The strip chart recorders were attached to output tachometers in a PUMA 500 control system and data was taken for VAL speeds 10 to 130. This data was used to estimate the velocity and acceleration for each joint. This method provided time estimates with an 80% accuracy for both straight line moves and joint interpolated moves. No specific information was given as to total cycle time estimate accuracy.

Tanner's Rule of Thumb, the RTM Method, Ahmadi's Method, and Steiner's Method are examples of cycle time estimates developed without knowledge of the control characteristics. They had varying degrees of accuracy. Extremely accurate

cycle time estimates would be provided if the detailed characteristics of the control system were known. SIM/7535 is an example of a simulation package developed using the control characteristics.

SIM/7535

IBM developed SIM/7535 for the IBM 7535 SCARA Robot using the control system characteristics [17]. This package provides cycle time estimates within ±1%.

The time required for an individual move was equal to the maximum of the time required for the three joints individually. Each joint was characterized by a time function based on move size. The timing functions were determined by robot kinematics and the control algorithms. The time function for large angles was a linear function and the time function for the small angles was proportional to the square root of the angle change.

2.2 SUMMARY

Accurate robot cycle time estimates are a valuable tool for increasing manufacturing flexibility by minimizing the time and effort required when installing a new robot system or a task changeover. Precise estimates are critical, particularly when integrating the robot with hard automation. Available simulation packages currently do not have the

ability to estimate cycle times accurately. This inaccuracy is due primarily to lack of knowledge of the detailed characteristics of the control system.

A number of empirical methods were developed based on discrete time estimates for elemental moves. These estimates provided cycle time estimates with varying accuracy levels. Steiner's time estimates, based on velocity and acceleration, provided time estimates with only 80% accuracy. In sharp contrast SIM/7535, based on the robot control characteristics, provided extremely accurate cycle time estimates (within $\pm$1%).

This literature search confirmed the need for a precise robot cycle time estimation method for an unknown control system.

CHAPTER 3: ESTIMATING IBM 7547 SCARA ROBOT CYCLE TIME

The investigation into robot cycle time estimation began as a result of an IBM-KSU research project. The project required the development of a method to estimate accurately the cycle time of an IBM 7547 SCARA robot and the development of an emulator to verify this method.

3.0 IBM 7547 SCARA ROBOT DESCRIPTION

The 7547 has three main components: the manipulator, the controller, and the control panel. The manipulator and controller are of primary interest when estimating robot cycle time. The IBM 7547 Manufacturing System Hardware Library describes the controller and manipulator:

"The controller contains most of the electronics for control of the manipulator operation. A microprocessor coordinates the manipulator's movement and monitors its speed and positioning.

"The manipulator is a two-jointed arm structure with four degrees of freedom. The joints of the arm, called Theta 1-axis and Theta 2-axis, provide 2 degrees of freedom through their swivel motion. The end-of-arm rotation, called the roll axis, provides a third degree of freedom. The end-of-arm also provides a fourth degree of freedom through a vertical shaft called the z-axis."

The manipulator has two DC servo motors which control axis movement. These DC servo motors are pulse driven. Pulses sent by the controller to the manipulator determine the robot's movement. The controller transforms the Cartesian coordinates within the AML program into pulses. These pulses tell the DC servo motors how far and at what speed to rotate a joint.

Understanding how the controller coordinates joint movement is key to estimating accurately the robot cycle time. The manuals provided by the manufacturer did not discuss this issue. Therefore a series of tests were run to allow the author and her colleagues to become more familiar with the 7547 SCARA Robot.

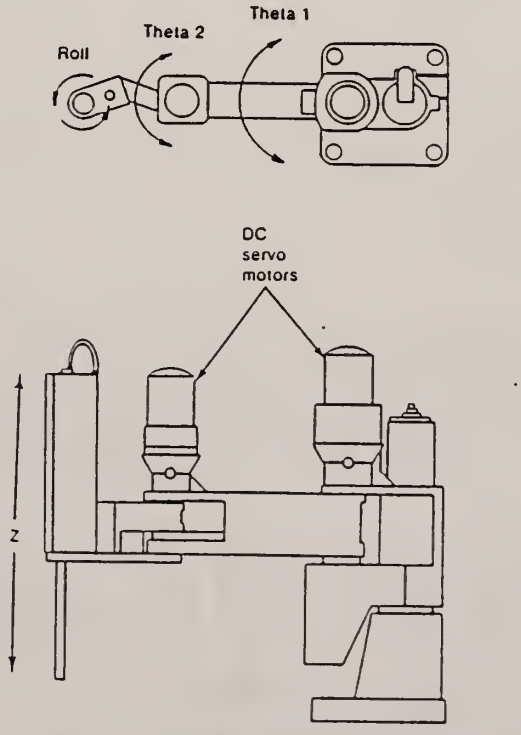


Figure 3.1: IBM 7547 SCARA Robot

3.1 STEP 1: EXPLORING THE 7547 SCARA JOINT MOVEMENT

The Theta axes were considered separately from the z-axis because the z-axis is operated independently of the Theta axes for material safety purposes. The z-axis is discussed in Section 3.6.

A series of timed movement tests were performed to analyze the robot arm movement. These tests provided data concerning the robot's speed at various locations within the work envelope.

Test Methodology - distance tests.

Tests were performed at the highest precision level. A linear path was not specified; the default "arc" path was used. The tests were timed using a stopwatch with accuracy of hundredths of a second. The same AML program was used in each test (with the points changed) for consistency. Seven locations were tested with all tests for each location beginning from the same point. These locations are shown in Figure 3.2.

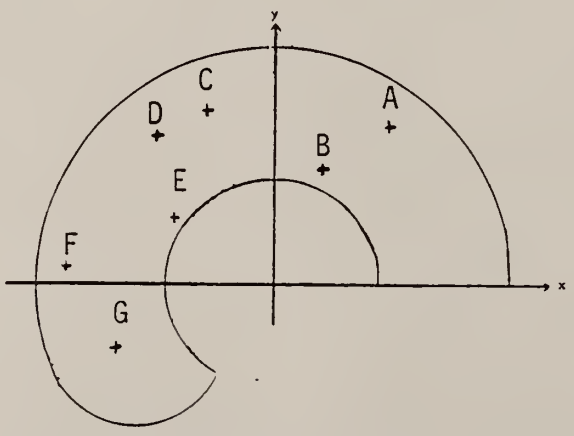
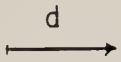


Figure 3.2: Work Station Test Locations

Tests were performed at each location for the following directions and distances.

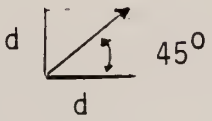
Directions:

x-only



y-only

xy



The distances tested were: 30 mm, 100 mm, 250 mm, 500 mm. The test consisted of the robot moving back and forth between two points 100 times. The robot moved from a test location (see Figure 3.2) to a point in a specific direction. The tests were replicated and averaged. The resulting data are included in the Appendix.

The data indicated that average robot speed (mm/sec) increased as distance traveled increased. The data also revealed speed variations throughout the work envelope. These tests led to the development of the following controller hypotheses.

3.2 THE CONTROL PROGRAM HYPOTHESES

Hypothesis 1: IBM 7547 SCARA uses joint interpolation.

During the distance tests it was observed that all joint

movement occurs simultaneously such that the joints begin and complete their movement at the same time. This led to the belief that the controller determines the speed for each joint resulting in a coordinated move (also called joint interpolation). The data from further tests agreed with this hypothesis, implying the controller uses joint interpolation.

Hypothesis 2: Robot speed is dependent upon the angular movement of each joint (elbow and shoulder).

The initial distance tests revealed distances alone were not useful for estimating movement times. This observation led to the hypothesis that robot speed depends on the average angular velocity of each individual joint and the corresponding angular displacement. This hypothesis was tested and verified by running a series of simple point to point moves and estimating average speeds based on angular displacement. (These tests are discussed in Section 3.3)

Hypothesis 3: The controlling joint is the slowest joint for a particular move.

Hypothesis 3 is a result of Hypotheses 1 and 2. The controlling joint for a particular move determines the time required for that move. This was verified by the distance tests. They demonstrated that movement was faster at the perimeter of the envelope and in the second quadrant. Both

areas maximize the use of the Theta 1-axis. This hypothesis was used to estimate the movement time for the original distance tests. The resulting data agreed with the hypothesis.

3.3 STEP 2: ESTIMATING JOINT SPEEDS

The Robotics Assembly Institute Handbook provided maximum tool tip speeds for each axis:

AXIS	MAXIMUM TOOL TIP SPEED
Theta 1	1400 mm/sec
Theta 2	850 mm/sec
Roll	450 degrees/sec
z	280 mm/sec

Figure 3.3: IBM 7547 SCARA Robot Tool Tip Speeds

Acceleration and deceleration effect the time required to perform a move. For many moves the tool tip will not reach the maximum speed. The arm will accelerate but never achieve the maximum speed before it begins to decelerate. Other moves may reach the maximum tool tip speed and remain at that speed for some period then decelerate to a stop. It would be very difficult to quantify this process retroactively. Therefore the time required for a move was estimated, based on the average speed of that move for a class of moves.

Joint Tests.

Each joint was evaluated by running a series of timed tests. The test consisted of having the robot perform a simple point to point move using one joint. The point to point move for Theta 1-axis was determined by evaluating this simple trigonometric relationship:

$$x = (1030)\cos\theta_1 \quad \text{and} \quad y = (1030)\sin\theta_1$$

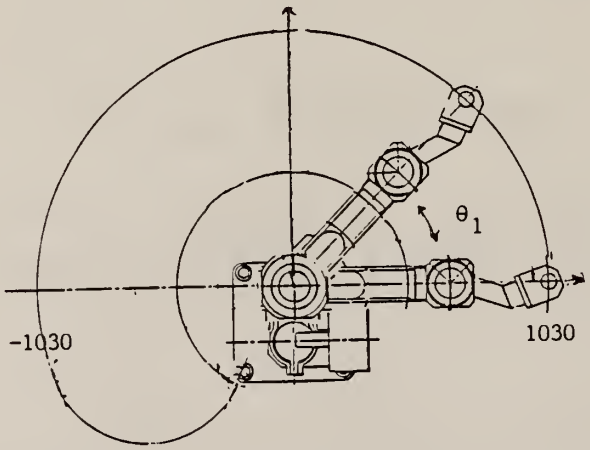


Figure 3.4: Theta 1 Joint Tests

Similarly for the Theta 2-axis:

$$x = 630 + (400)\cos\theta_2 \quad \text{and} \quad y = (400)\sin\theta_2$$

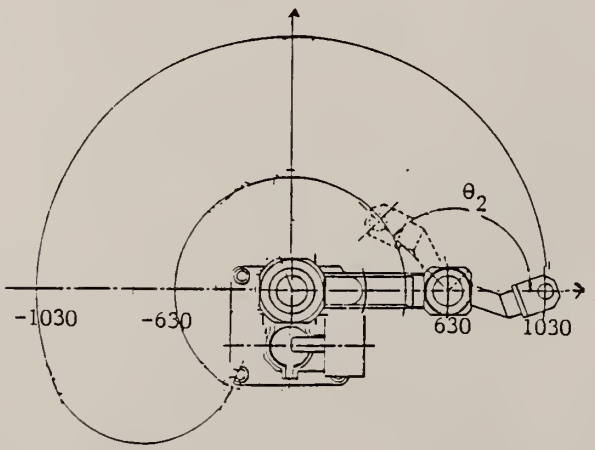


Figure 3.5 Theta 2 Joint Tests

The joint tests ranged from 0.01 radians to the joint's maximum (Theta 1-axis maximum = 3.49 radians, Theta 2-axis maximum = 2.79 radians). These tests were performed at the highest precision level and with the controller set for medium speed. A list of the tests are included in the Appendix.

The robot cycled 50 times for each test and the time required was recorded. The tests were replicated and the times were averaged. The average time was used to determine the speed for that move. For example, in the test of θ_1 = 1.1 radians, the total radians traveled in 50 cycles = $\theta_1 100$ = 110 radians. The average time for the series of tests equals 169.7 seconds. Thus the speed was equal to 110 radians/169.7 seconds = 0.648 radians/second. The average speed was computed for each test.

The angular moves tested (1.1 radians in the example above) and the resulting average speed (0.648 rad/sec) were the data analyzed to characterize the robot movement. (The appendix contains these data.)

3.4 STEP 3: REGRESSION ANALYSIS

A simple regression analysis using the average speed and angular displacement for each test was performed using the SAS statistical software package. The purpose of the regression analysis was to develop equations for accurately estimating the joint's average speed for a specific angular

move. The resulting time estimate must be within ±5% (IBM-KSU research project specification) of the actual time required to perform the move. The time estimate was calculated by dividing the angle moved by the average speed estimate.

The equations of motion (acceleration, velocity and distance) are typically in quadratic form and the regression equations were expected to be of that same form. The initial distance test data were used for testing the regression equations. One regression equation (for each joint) could not adequately describe (within specifications) the average speed for the joint. This is logical, due to acceleration and deceleration of the robot arm.

Regression equations were developed and tested using a variety of data subsets. The initial data set consisted of less than 10 points per joint distributed throughout the work envelope. Developing the data was basically a trial and supplement process. More joint tests were added to the data set when the regression equations were not within the ±5% accuracy specification.

The final data set consisted of joint tests for nearly every 0.1 radian increment. The exception to the 0.1 radian system was for very small moves, less than 0.1 radians. This region was tested in increments of 0.01 radians. (Joint tests are detailed in the Appendix.) The appropriate data

subsets were determined by separating the data, developing the regression equation for the data subsets, then testing the accuracy of the resulting equation. This procedure was repeated until the data subsets met the accuracy requirements of $\pm 5\%$. The regression equations were truncated:

For the Theta 1-axis (θ_1 radians):

$$\text{Speed} \begin{cases} = .0134 + 1.6750\theta_1 - 3.7410\theta_1^2 & \theta_1 \leq 0.1 \\ = .0588 + .9387\theta_1 - .3804\theta_1^2 & 0.1 < \theta_1 \leq 1.0 \\ = .2977 + .4014\theta_1 - .0757\theta_1^2 & 1.0 < \theta_1 \leq 2.0 \\ = .4723 + .2176\theta_1 - .0273\theta_1^2 & 2.0 < \theta_1 \leq 3.0 \\ = .9587 - .0981\theta_1 + .0240\theta_1^2 & 3.0 < \theta_1 \leq 3.49 \end{cases}$$

For the Theta 2-axis (θ_2 radians):

$$\text{Speed} \begin{cases} = .0125 + 1.9360\theta_2 - 4.0260\theta_2^2 & \theta_2 \leq 0.1 \\ = .0690 + 1.0720\theta_2 - .3504\theta_2^2 & 0.1 < \theta_2 \leq 1.0 \\ = .2928 + .6107\theta_2 - .1079\theta_2^2 & 1.0 < \theta_2 \leq 2.0 \\ = .4733 + .4111\theta_2 - .0528\theta_2^2 & 2.0 < \theta_2 \leq 2.79 \end{cases}$$

These equations give the average speed reached by a joint for a specific move (in radians/second). The regression equations developed for each subset are described in more detail in the Appendix.

3.5 STEP 4: DEVELOP CYCLE TIME ESTIMATE ALGORITHM

The regression equations are used to determine the

average speeds reached by each joint for a specific point to point move. These speeds are used to calculate the time required by each joint for the move. Then these times are compared and the slowest time is used as the time estimate for that move. The slowest time estimate is applied because the controller coordinates the joint movement so that the joints begin and end movement simultaneously. Therefore the move can be completed only as quickly as the slowest joint. The logic is discussed in the following section.

Logic For Time Estimate Of a Simple Point to Point Move

Given a current position (x_1,y_1) and a goal position (x_2,y_2), the first task is to convert the Cartesian coordinates into polar coordinates. The reference point used is the home position of (1030,0).

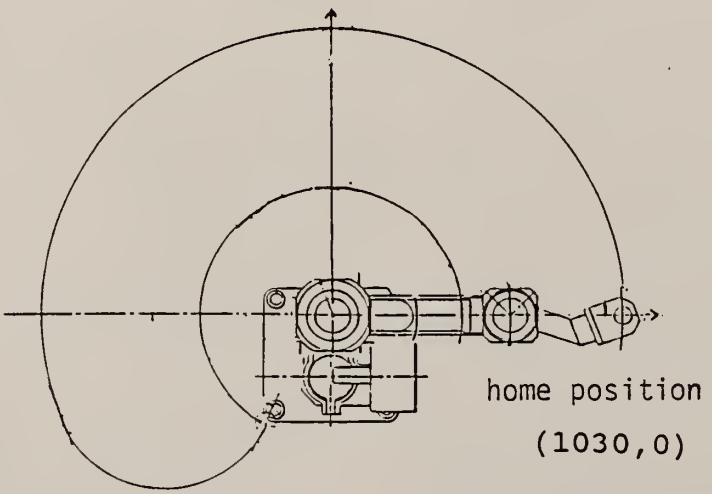


Figure 3.6: Work Envelope

The 7547 has the ability to move in three quadrants. The logic is displayed for each quadrant.

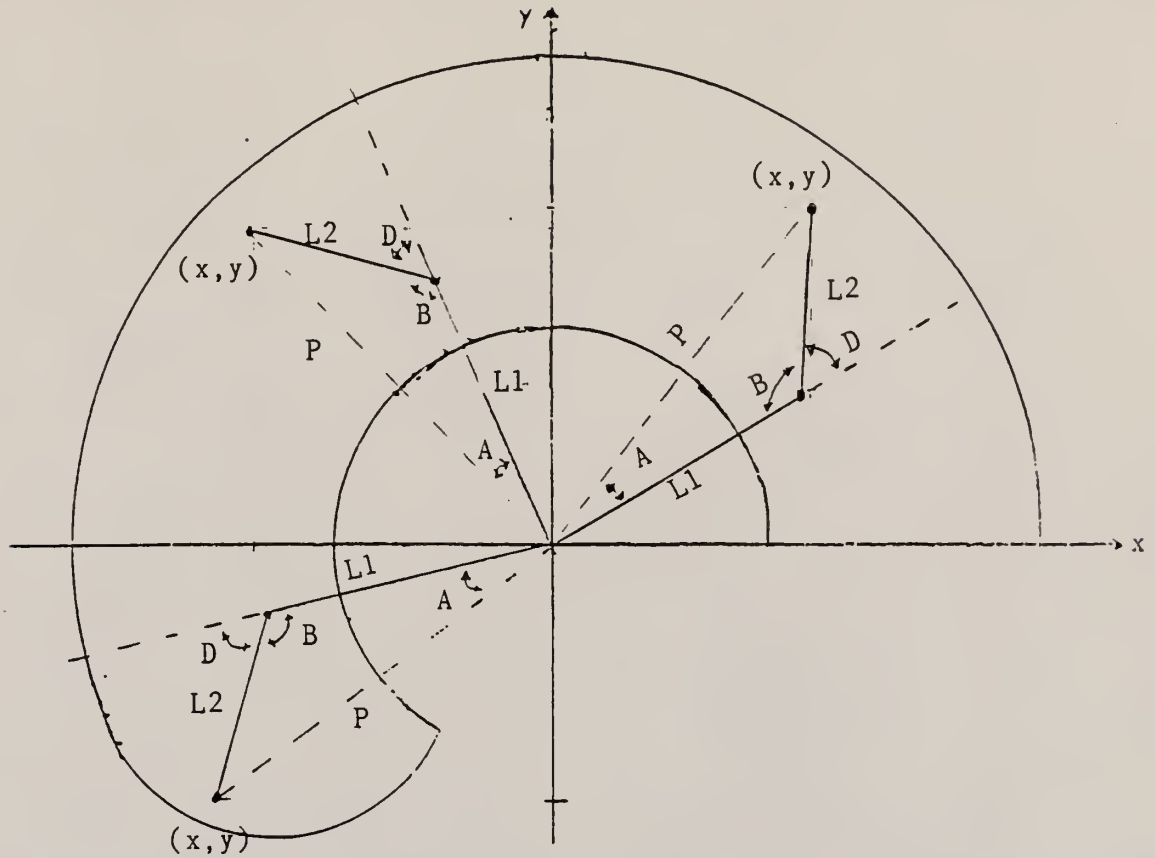


Figure 3.7: Time Estimate Logic

$$P = \sqrt{x_i^2 + y_i^2} \qquad (1)$$

The length of Theta 1-axis is 630 mm and is indicated as L1. The length of Theta 2-axis is 400 mm and is indicated as L2.

By the law of cosines:

$$B = \cos^{-1}\left(\frac{L1^2 + L2^2 - P^2}{2(L1)(L2)}\right) \qquad (2)$$

By the law of sines:

$$A = \sin^{-1}\left(L2 \frac{\sin B}{P}\right) \tag{3}$$

$$D = \pi - B \tag{4}$$

where D is the movement for the Theta-2 axis from the zero position.

$$F = \tan^{-1} \frac{|y_i|}{|x_i|} \tag{5}$$

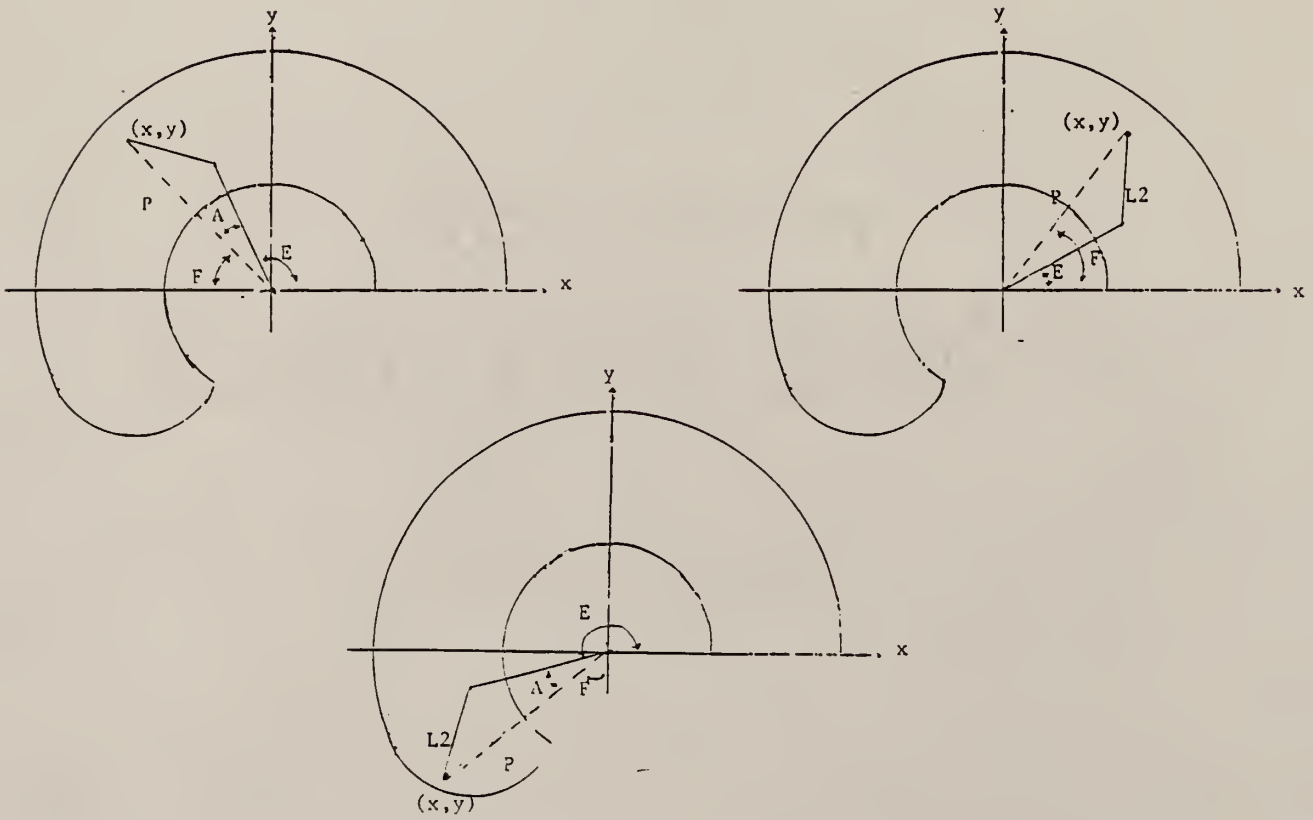


Figure 3.8: Time Estimate Logic

E is defined to be the angular movement for the Theta 2 axis from the zero position and is quadrant dependent.

$$E = |F| - |A| \quad \text{for} \quad x > 0 \quad \text{and} \quad y > 0 \tag{6a}$$

$$E = \pi - |F| - |A| \quad \text{for} \quad x < 0 \quad \text{and} \quad y > 0 \tag{6b}$$

$$E = (3/2)\pi - |F| - |A| \quad \text{for} \quad x < 0 \quad \text{and} \quad y < 0 \tag{6c}$$

The conversion process is repeated for each (x_i, y_i).

The primary angles of interest are D_i and E_i. Once these are computed, the movement required for each axis is determined.

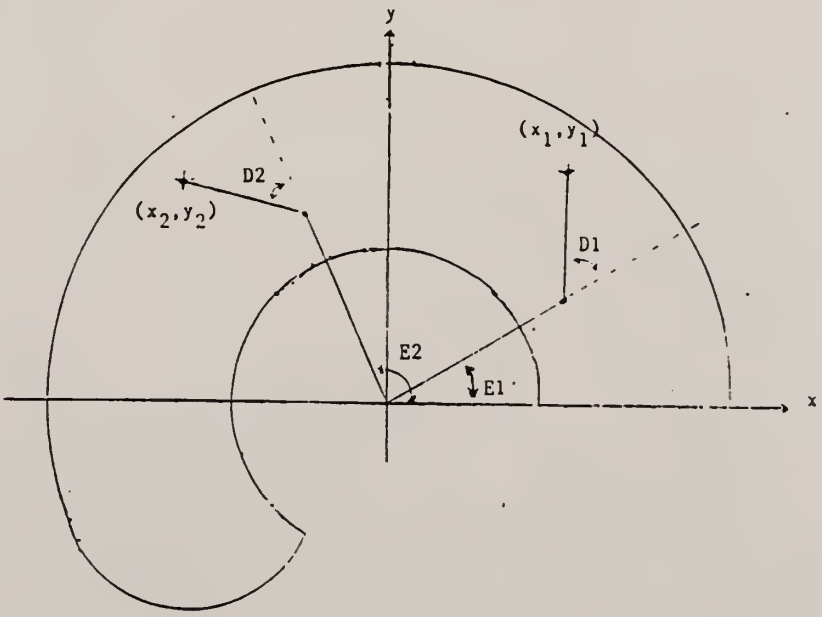


Figure 3.9: Angular Movement

$$\theta_1 = D_1 - D_2 \tag{7}$$

$$\theta_2 = E_1 - E_2 \tag{8}$$

This gives the angular move the manipulator axes must make to perform the point to point move.

Now the appropriate speed equation is employed and estimates for the average speeds are given, $\hat{S1}$ and $\hat{S2}$. These are converted into time estimates:

$$\frac{\theta_1}{\hat{S1}} = \hat{T1} \tag{9}$$

$$\frac{\theta_2}{\hat{S2}} = \hat{T2} \tag{10}$$

The maximum time value is chosen as the time estimate for the point to point move. This maximum time value is the time required for the slowest joint to complete the move.

$$MAX(\hat{T1},\hat{T2}) = \hat{T}_{1,2} \tag{11}$$

3.6 END OF ARM MOVEMENT: Z-AXIS

The z-axis was evaluated by running a series of timed tests very similar to the timed joint tests discussed previously. The tests consisted of the robot performing a simple z move (no joint movement). These tests were performed at the highest precision level and with the controller set for medium speed. The z move tests ranged from -10 mm to -250 mm in 10 and 20 mm increments. A list of the tests are included in the Appendix. The tests were

replicated and the times averaged. The average time was used to characterize the speed for that move.

The z moves tested and the resulting average speeds were the data analyzed to characterize the z-axis movement. A regression analysis using the SAS statistical software package was performed. The following is the truncated regression equation developed:

$$\text{Speed} = 12.5900 + 0.9364Z - 0.0018Z^2 \quad (Z \text{ mm})$$

The average speed can be used to calculate the time required to perform a specific z move. However, in most applications a discrete time estimate must also be added to account for grasp, release and any programmed delays.

For the IBM-KSU Research project application of the 7547 SCARA Robot, the z-axis was timed for the moves required in the mock-up. The timed moves included grasps, releases and delays. A list of moves and times are included in the Appendix. These times were incorporated into the emulator and are added to the cycle time.

3.7 STEP 5: THE EMULATOR PROGRAM

The emulator program was developed to verify the characterization of the IBM 7547 SCARA Robot. The emulator is also a tool for estimating the time required for the robot to move through a pick and place sequence.

The 7547 emulator uses the current set-up, error handling routines, and a form of the bill of materials for a given plate to develop the pick and place sequence and to estimate the cycle time. Time estimates are determined for each discrete move and are added together to form the cycle time estimate.

The robot layout is shown in Figure 3.10.

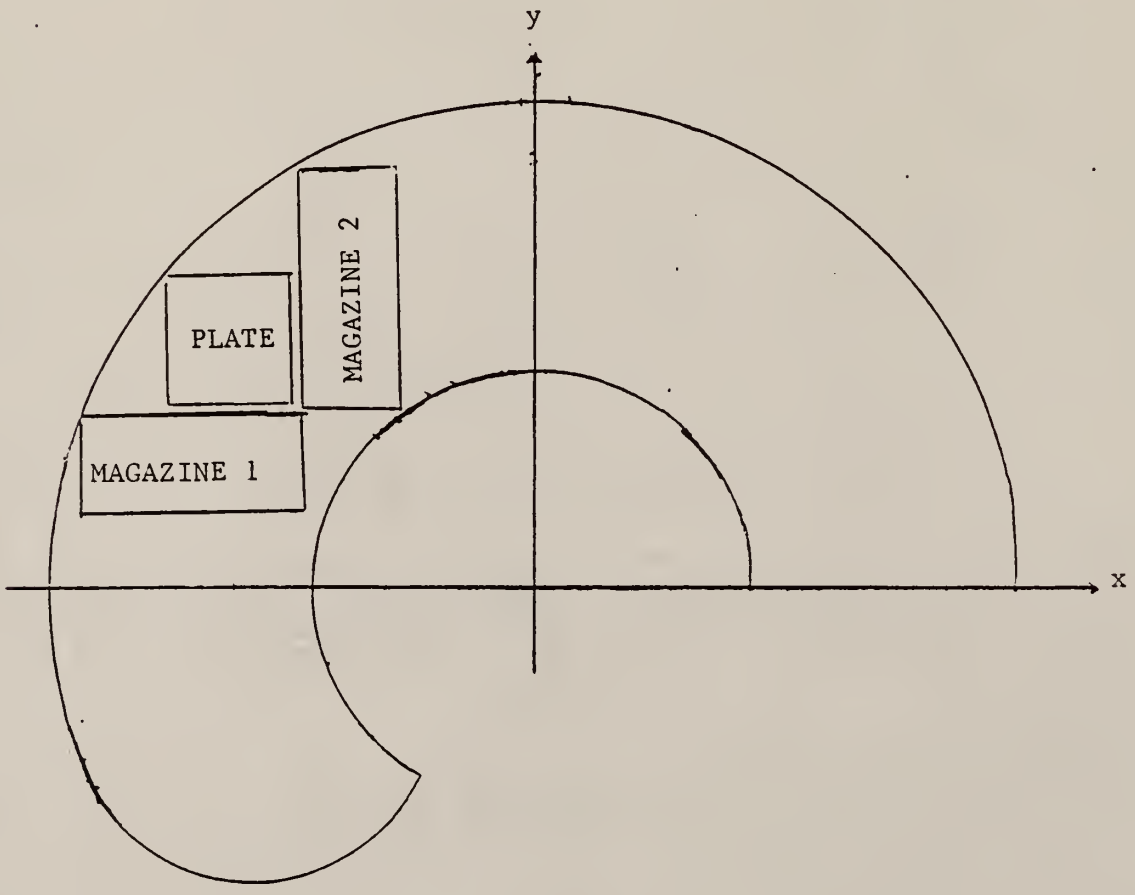


Figure 3.10: Work Station Layout

The robot moves along a predetermined path when performing the pick and place sequence. The bill of materials (BOM) is used to develop the path the robot follows. The pick and place sequence consists of looping through the following moves:

1: MOVE to empty dispenser

2: PICK UP empty dispenser

3: MOVE to the cartridge

4: PLACE the dispenser in empty space within the cartridge

5: MOVE to next closest full dispenser

6: PICK UP full dispenser

7: MOVE to initial location (on plate)

8: PLACE the full dispenser

Figure 3.11: Robot Moves

(See Figure 1.1.)

The 7547 emulator uses an appended version of the BOM file to determine the cycle time required. This version is sorted into the desired internal plate path and contains each dispenser's preferred cartridge choice for each magazine. An example of the appended BOM file is shown below.

Part Type	Plate Location: X	Y	Preferred Cartridge for: Magazine 1	Magazine 2
1	-676	625	2	4

Figure 3.12: BOM Appended Version

The work station design includes two magazines. The robot is programmed to pick from one magazine until a part type is depleted. Then the robot picks from the alternate magazine. Cartridges are replenished during plate loading to minimize downtime. When a finished plate is removed and a new plate loaded, any empty or near-empty cartridges are removed and replaced. Thus the emulator design allows the operator to specify the magazine status and to specify the cartridge replenishment scheme. The operator may elect to replace cartridges when the cartridge contains 0, 1, 2, or 3 dispensers. Those cartridges within the current magazine being used are replaced automatically when the emulator switches to the alternate magazine. No time is added to the cycle time estimate for replacement of cartridges. When the emulator switches magazines a screen message notifies the operator.

The emulator incorporates multiple plate runs to evaluate the effect of various magazine states. The cycle times developed for each run are written to a file for future analysis. This feature is useful when evaluating various path heuristics and potential cartridge replenishment schemes.

Figure 3.13 is a flowchart of the emulator.

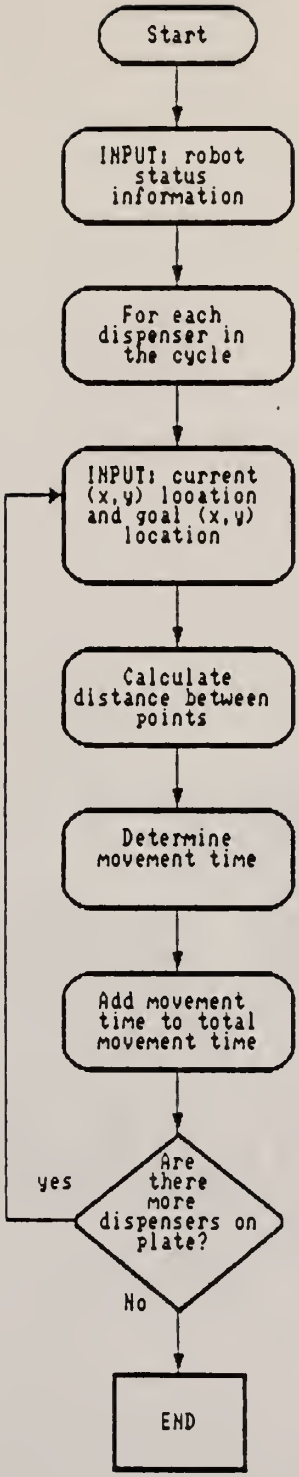


Figure 3.13: Emulator Flowchart

3.8 STEP 6: VERIFY EMULATOR

The emulator was tested to verify magazine status after each cycle. Plate 5 was tested for five cycles. The test data are included in the Appendix.

Next the emulator was tested to determine how variations in payload would effect the cycle time estimates. The payload tests were conducted with modified versions of the emulator and the original AML program. These modified versions considered only joint moves (no z moves, grasps, releases, or delays).

The modified emulator was first tested against the modified AML program with no payload and the emulator consistently underestimated the actual cycle time. See Figure 3.14.

	No payload Time (seconds)			
Run	Actual	Emulator Estimate	Difference	Percent Error
1	125.36	110.01	-15.35	-12.2
2	121.91	106.48	-15.43	-12.6
3	132.59	115.45	-17.14	-12.9
4	127.11	110.14	-16.97	-12.8
5	124.62	109.63	-14.99	-12.0

Figure 3.14: Modified Emulator & AML Program

The search for the reason why the emulator consistently underestimated actual cycle time began with the joint regression equations. The joint regression equations were verified with additional data. (See the Appendix.) The 7547 emulator program was error tested for programming errors. None was found. Additional series of point to point moves were tested against the 7547 emulator estimates and they proved very accurate (within $\pm$1%). The next area investigated was the AML program. The IBM 7547 SCARA Robot uses AML, a very inflexible program. In order to handle the specific requirements of the mock-up (see Figure 1.1) a large number of counters were used. (The AML program used is included in the Appendix.) The AML program used must be affecting the time requirements. To test this theory the identical pick and place sequence developed by the AML program was also programmed into a simpler version of strictly pmoves (defined point to point moves). The results were:

AML PMOVE PROGRAM TIME	MODIFIED AML PROGRAM ACTUAL TIME	MODIFIED EMULATOR ACTUAL ESTIMATE
110.19 secs	125.36 secs	110.01 secs

The extra time required by the AML program must be used to evaluate the various counters used to keep track of the magazine states.

The system was evaluated for two additional payload conditions: 8 lbs and 16 lbs. Figure 3.15 shows that these payloads produced slight degradations, but had no real effect on the system performance.

Modified AML Program

	Actual Time (seconds)		
PLATE	0 lb	8lb	16 lb
1	125.36	125.20	125.25
2	125.16	126.01	125.60
3	124.02	124.30	124.52
5	124.42	124.43	124.69

Figure 3.15: Time for Various Payloads

The z-axis was also tested independently and showed no time variations. More payload data is included in the Appendix.

Finally, the emulator was tested to verify the accuracy of the cycle time predictions. This version includes all z moves, grasps, releases and delays. The emulator was accurate within $\pm$1% of actual cycle time. (See Figure 3.16.) When the z-axis, grasps, releases and delays were included, the emulator was very accurate. The improved accuracy is due to the inclusion of delays. The z-move estimates include a 0.5 second delay before each grasp and release that was programmed into the original AML program. This 0.5 second delay is used by the controller to evaluate the counters.

Plate	Cycle	Time (seconds) Estimate	Actual	Percent Error
1	1	352.8639	352.53	0.1
	2	349.3338	349.77	-0.1
2	1	353.2606	352.54	0.2
	2	349.3904	349.20	0.0
3	1	351.5479	352.00	-0.1
	2	347.7172	348.45	-0.2
5	1	351.6333	352.35	-0.2
	2	348.4996	349.64	-0.3

Figure 3.16: Emulator Tests

3.9: SUMMARY

This chapter presented the process developed to estimate precisely the cycle time of an IBM 7547 SCARA Robot. The estimate was developed as partial fulfillment of an IBM-KSU Research project requirement. In addition, an emulator was developed which verified the characterization and estimated cycle time within $\pm$1% for the pick and place application. The 7547 emulator program verified the hypotheses developed:

1) IBM 7547 SCARA Robot uses joint interpolation.
2) Robot speed is dependent upon the angular movement of each joint (elbow and shoulder).
3) The controlling joint is the slowest joint for a particular move.

The 7547 emulator was shown valid for a variety of payloads. The modified 7547 emulator demonstrated potential cycle time estimation problems when using a large number of counters in the AML program. This implies that cycle time estimation must consider the robot program and programming efficiency.

CHAPTER 4: A GENERAL METHOD FOR ESTIMATING ROBOT CYCLE TIME

This chapter presents a general method to estimate precisely robot cycle time without knowledge of the control system. Current literature confirms the importance of estimating robot cycle time; however, the literature does not contain an accurate method. The method described herein was generalized from the process applied to estimate the cycle time of an IBM 7547 SCARA Robot pick and place application.

4.0 THE METHOD

Using this method robot cycle time estimates can be developed in a quick and simple fashion. The robot control characteristics are unknown, so the robot is treated as a black box. The method consists of six steps.

STEP 1: Determine how joint movement is coordinated.

Understanding how the controller coordinates joint movement is key to developing an accurate cycle time estimate. In general, robots move in one of the following three modes or a combination of these modes (20):

1. Straight line: The end of arm tool follows a straight line from its current position to the next point at the current robot speed.
2. Joint interpolation: Speed of each joint is prorated so that all joint motions terminate simultaneously.

3. Slew: All joints move at maximum speed. Some joints complete the move before others.

The motion mode used by the robot will effect the algorithm developed to estimate the cycle time so the robot must be studied to identify which motion mode(s) is(are) applicable. The study consists of running a series of timed tests and observing the robot motion. After these tests are completed, develop hypotheses to describe the motion mode.

The slew and joint interpolation modes result in the same hypothesis: the controlling joint is the slowest joint for a particular move. In other words, the controlling joint for a particular move determines the time required for that move. The straight line mode requires a different hypothesis; one based on a prescribed robot speed.

STEP 2: Develop tests to describe average axis speed.

Manufacturers provide machine specifications which include idealized process rates and maximum tool tip speeds for each axis. These specifications do not provide enough information on which to base an accurate cycle time estimate. For many moves the tool tip will not reach the maximum speed. The arm will accelerate but never achieve the maximum speed before it begins to decelerate (as shown in Figure 4.1). Other moves may reach the maximum tool tip speed and remain at that speed for some period then decelerate to a stop (as

shown in Figure 4.2).

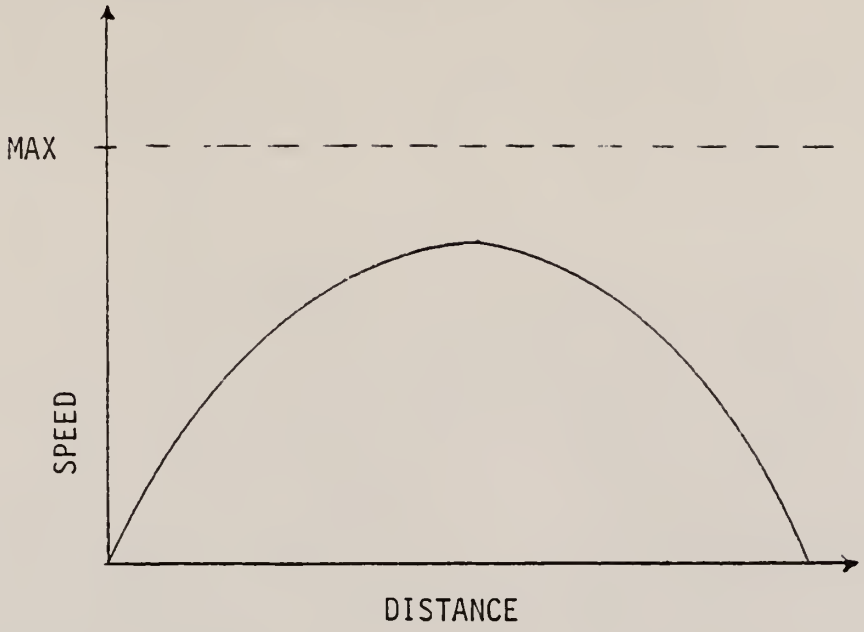


Figure 4.1: Maximum Tool Tip Speed Not Reached

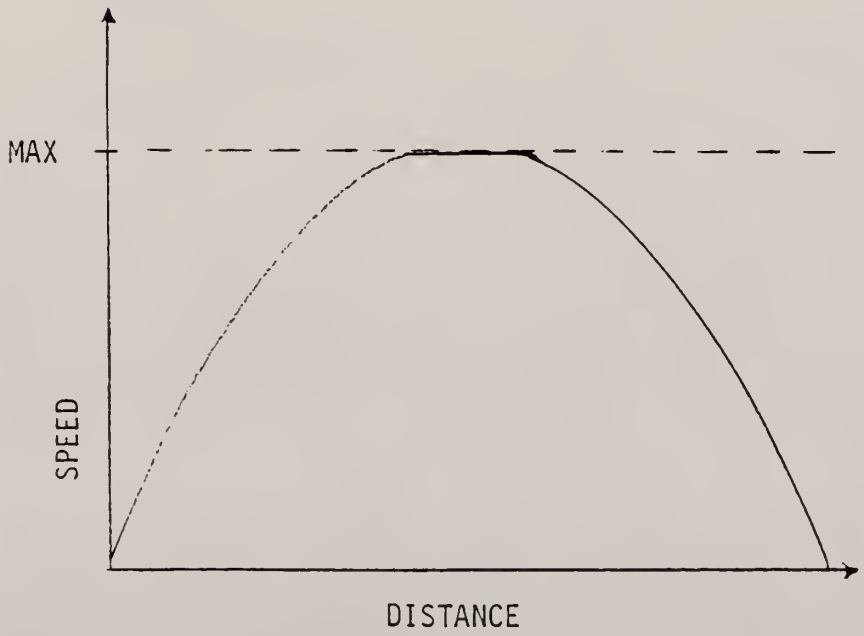


Figure 4.2: Maximum Tool Tip Speed Reached

It is very difficult to quantify this process retroactively. Therefore the time required for a move is estimated based on the average speed of that move.

Each axis is evaluated by running a series of timed tests. The tests should consist of the robot performing a simple move using only one axis. The axes tests should range from the smallest to largest possible angular move. Replicate the tests and average the times. Convert this average time data into average speed for different angle categories. (Divide the angular move by the average time.)

STEP 3: Perform a regression analysis.

The purpose of the regression analysis is to develop equations which provide an accurate estimate of the axis' average speed for a specific angular move. The regression analysis uses the average speed and angular displacement for each test. Test the resulting equations to determine the accuracy of the predictions. This is a trial and supplement process. Additional average speed data may be required to meet the prescribed accuracy requirements. Thus steps 2 and 3 may be repeated several times until the required precision is achieved.

STEP 4: Develop a cycle time estimate algorithm.

Use the regression equations and the control program hypotheses (developed from motion modes) to develop an

algorithm which provides the time estimate for a simple point to point move. The algorithm will be the basis for developing the emulator. Test the algorithm and validate the hypotheses.

Step 5: Develop an emulator.

The robot emulator is a software characterization of the robot's processes. The emulator's purpose is to provide an accurate cycle time prediction for a given layout and move sequence. The emulator will use the algorithm developed, current layout and move sequence to verify the cycle time estimation method. When developing the emulator, programmed delays and programming efficiency must be considered.

Step 6: Verify emulator accuracy.

Emulator accuracy is verified by comparing the emulator time estimates to actual time required for various move sequences. Variations in payloads and controller speed switches for the particular application must be considered for accurate estimates.

4.1 SUMMARY

This chapter presented a general method for precisely estimating robot cycle time in a quick and simple fashion. This method consists of six steps:

1. Determine how joint movement is coordinated.

2. Develop tests to describe average axis speed.

3. Perform regression analysis to develop equations which describe average axis speed.

4. Develop a cycle time estimate algorithm.

5. Develop an emulator

6. Verify emulator accuracy.

The utility of this simple six step method lies in its lack of reliance on the control system characteristics and its accuracy.

CHAPTER 5: CONCLUSION AND FUTURE CONSIDERATIONS

5.0 SUMMARY

This thesis presented a general method to estimate precisely robot cycle time without knowledge of the control system.

The research into robot cycle time estimation began as a result of an IBM-KSU research project. The research project required a method to accurately estimate the 7547 SCARA Robot cycle time (within $\pm$5%) and an emulator to verify this method.

A literature search confirmed the value of robot cycle time estimates as an important planning tool. Cycle time estimates are used to evaluate various robots, test the layout and test the path choice. These estimates are also useful to determine system capacity, balance production and establish schedules.

An established empirical method for estimating precisely robot cycle time was not found in this search. Varying degrees of accuracy were provided by the empirically based methods presented in the literature. The inaccuracy is due to manufacturing process variations, mass of the payload, and/or an inaccurate characterization of the control system. Manufacturers will not release the control system information, making precise modeling very difficult.

To complete the IBM-KSU research project the IBM 7547 SCARA Robot was characterized empirically. An emulator was developed to verify the characterization. This provided cycle time estimates within ±1%, which favorably compares to estimates developed for an IBM 7535 based on a known control system. From this process a generalized method was developed to estimate robot cycle times in a quick and easy fashion.

The method consists of six steps:

STEP 1: Determine how joint movement is coordinated.

STEP 2: Develop tests to describe average axis speed.

STEP 3: Perform a regression analysis.

STEP 4: Develop an algorithm using the hypotheses.

STEP 5: Develop an emulator.

STEP 6: Verify emulator accuracy.

5.1 FUTURE APPLICATIONS

The general method could be applied to characterize completely the IBM 7547 SCARA Robot. To do so would require considering all speed switch settings and precision levels and the emulator would also need to be expanded.

It would be interesting to see the general method applied to other classes of robots. An extension of this would be to link a number of these robots and estimate a group cycle time.

It would also be interesting to see how an emulator developed for one specific robot would work on another robot

of that same type, i.e. how does manufacturing process variation effect the characterization? If the characterization is accurate across a group of robots, the general method could be successfully applied to simulation packages.

5.2 FUTURE RESEARCH

The following areas are recommended for future research:

1. Develop an emulator to use the actual robot program as input and then develop the cycle time estimate.
2. Minimize the number of tests required to estimate axis average speed.
3. Develop a way to automate the axis tests.
4. Develop the control characteristics using the emulator data.

REFERENCES

1. Ahmadi, J. Proposal for development of optimized control and support programs for IBM 7547 SCARA Robot used for PAPS automated plate loading task, April 15, 1988.

2. Ahmadi, J., S. Grotzinger and D. Johnson. "Component allocation and partitioning for a dual delivery placement machine," Operations Research, vol 36, no 2, March-April 1988, 176-191.

3. Ahmadi, J., S. Grotzinger and D. Johnson. Research Report: Emulation of concurrency in circuit card assembly machines, IBM: RC12161, revised July 7, 1988.

4. Anonymous, "Measuring robot motion aids selection process," Robotics World, May 1987, 22-24.

5. Behringer, C.A. "Robots in assembly - if you've got the time," Manufacturing Engineering, June 1987, 45-48.

6. Chang, Chi-Ming. "On optimizing assembly plans for a robotized printed circuit board assembly center." Doctoral Thesis, Texas Tech University, December 1986.

7. Cranmer, L., L. Satzler, and S. Shridar. "Final Report to IBM: Development of optimized control and support programs for IBM 7547 SCARA Robot used for PAPS automated plate loading task," Kansas State University, 1989.

8. Groover, Mikell P. Automation, production systems and computer integrated manufacturing, Prentice Hall, New Jersey: 1987.

9. Heginbotham, W., M. Dooner and K. Case. "Robot application simulation," The Industrial Robot, June 1979, 76-80.

10. Heginbotham, W., M. Dooner and D. Kennedy. "Analysis of industrial robot behaviour," Third CISM-IFTOMM Symposium, 1980, 434-446.

11. Hershey, R., A. Letzt, and S. Nof. "Computerized methods for predicting robot performance," SME Technical Paper, AUTOFACT 5 Conference, November 14-17, 1983.

12. Hornick, M. "Off-line robot programming and simulation," Doctoral Thesis, University of Wisconsin-Madison, May 1985.

13. Iversen, W. "This just may speed the use of robots," Electronics, October 15, 1987, 32.

14. Krepchin, I. "Robots: plan now - pay off later," Modern Materials Handling, April 1987, 91-94.

15. Kretch, S. "Robotic Animation," Computers in Engineering: Proceedings of the International Computer Engineering Conference 1982, 87-91.

16. Mattis, P. and K. Gill. "The best robot for the job: simulation can help decide," The Industrial Robot, March 1988, 32-34.

17. Meyer, J. and R. Jayaraman. "Simulating robotic applications on a personal computer," Computers in Mechanical Engineering, July 1983, 14-18.

18. McGalliard, G. "A general simulation program for robot manipulator arm dynamics," M.S. Thesis, Naval Postgraduate School, September 1984.

19. Nof, S. and H. Lechtman. "Robot time and motion system provides means of evaluating alternative robot work methods," Industrial Engineering, April 1982, 38-48.

20. Peck, B. "Robotic simulation," BDX-613-3410, Bendex Kansas City Division, February 1986.

21. Rahnejat, H. "Simulating for 'resource optimization' in robot assisted automatic assembly," Proc. Instn. Mech. Engrs., vol 200, no B3, 181-186.

22. Satzler, L. "Robot cycle time minimization in electronic assembly," M.S. Thesis, Kansas State University, 1989.

23. Schroer, B. and W. Tech. "A graphical simulation tool with off-line programming," Simulation, August 1986, 63-67.

24. Steiner, M. "Robots: practical application and simulation," Computers in Engineering: Proceedings of the International Computer Engineering Conference 1982, 101-105.

25. Tanner, W. "Can I use a robot?", Robotics Today, Spring 1980, 43-44.

26. Warnecke, H., R. Schraft, and V. Schmidt-Streier. "Computer graphics planning of industrial robot application," Third CISM-IFTOMM Symposium, 1980, 521-542.

27. Welch, T. "Evaluating robot systems through simulation," SME Technical Paper, 13th ISIR/Robots 7 Conference, April 17-21, 1983, Chicago.

APPENDIX

Proposal for Development of Optimized Control and Support Programs for IBM 7547 Scara Robot used for PAPS Automated Plate Loading Task

Javad Ahmadi
IBM Corporation
11400 Burnet Road
Dept. C0A/Bldg. 001
Austin, TX 78758
Phone: 512/823-7170

April 15, 1988

The following is a proposal for development of a set of models and programs for optimized planning and operation of the IBM 7547 Scara Robot selected for "plate loading" operation automation. It is not by any means a comprehensive specification, additional details will be provided at later stages.

Table of Contents

		Page
1.0	Plate Loading	1
2.0	The Plate Loading Robot	1
3.0	Programming	1
	3.1 Robot Loading	4
	3.2 Robot Setup	4
	3.3 Pick and Place Sequencing	4
	3.4 Step Queueing	5
4.0	Robot Usage Environment	5
5.0	Robot Emulation	5
6.0	Exception Handling	5
	6.1 Exception Detection and Recovery Requirements in the Controller	5
	6.2 Precision and Accuracy Requirements	5
7.0	Data Supply, User Interfacing, and Program Management	6

1.0 Plate Loading

Simply stated, plate loading is a pick and place operation that is easily accomplished manually. The task involves removal of vertically-mounted empty dispensers from a fixture plate and their replacement with full dispensers. Each dispenser is placed and held at a precise location on the plate with a given orientation. All dispensers are identical in shape and size, requiring the same handling. However, dispensers are part-type specific. A given part type may be utilized in several positions on the plate. Since there may be many dispensers and all identical looking, the manual plate loading becomes labor intensive and error prone. Hence, it is suitable for automation.

The fixture plate bill of materials specification (BOM) supplies the part types, their placement location, and orientation. A schematic of the plate fixture and dispensers is given in Figure 1.

Full dispensers are supplied in magazines. All magazines are identical and have capacity for a limited number of dispensers.

2.0 The Plate Loading Robot

An IBM 7547 Scara Robot is being fitted for automation of the dispenser loading task. A schematic of the design is given in Figure 2.

This diagram depicts two stationary magazine racks for positioning of dispenser magazines in discretely defined feeder slots. Also shown is the location and mounting of the plate fixture.

The robot operation consists of removal of an empty dispenser (if any) from its current location and its return to an empty slot within the appropriate type magazine. Then, a full dispenser from the same magazine is retrieved and placed at the same position with proper orientation. These steps are repeated for all dispensers on the plate. Loading and unloading of the plate, as well as replenishment of dispenser magazines is done manually.

Note that full dispensers and empty dispensers are held at opposite orientations in the magazine.

3.0 Programming

An IBM PC-based control program generator that can supply a given pick and place instruction sequence for operation of the robot is needed. The instruction set supplies the NC data consisting of location of each

feeder magazine, placement location and their orientation, and the precise handling and motion of the robot arm. The instruction sequence must be free of interference problems. The control program must be capable of exception detection and recovery actions whenever they occur. Additional details on capabilities of the control program will be supplied at a later time.

The sequential dependency of the pick and placement steps calls for optimization of the placement sequence. Purpose of the optimization is minimization of the robot cycle time.

A hierarchy of decision problems for optimized operation of the robot are assumed. These problems are described in the following sections.

3.1 Robot Loading

Generally, the robot magazine capacity is in excess of the part types (mix) required. It should, therefore, be possible to load the robot judiciously with multiple magazines of the same kind whenever possible. In this manner, operator intervention for replenishment of the robot can be minimized. This, of course, implies additional capability on the part of the robot controller in utilization of alternate magazines whenever specified.

3.2 Robot Setup

Clearly, positioning of the feeder magazines effects the robot's reach delay in accessing of the dispensers. Hence, the optimum location of each feeder magazine is needed. A program that will assist the operators in optimum setup of the tool will be required.

3.3 Pick and Place Sequencing

Depending on the positioning of the feeder magazines and placement location of the dispensers, and subject to interference restrictions, the optimum sequence of the operations is needed. This determination should be possible in two modes:

a. A given robot setup already exists.
b. No setup has been implemented and, hence, an integrated solution of the decision hierarchies may be possible.

3.4 Step Queueing

If during the execution of the steps an empty magazine is encountered, it is preferred to queue this step to another time. In the meanwhile the robot will work on steps dealing with opposite rack and hence permitting the operator to replenish the robot without stoppage.

4.0 Robot Usage Environment

Generally, a single robot will be used for support of a family of distinct plates that are under concurrent usage. The usage rates may be equal or vary according to certain demand ratios. In this case, the staging of the robot is based on an aggregated BOM that supports the family of plates under consideration. This may be an additional factor in design of optimization models for support of robot tasks.

5.0 Robot Emulation

The purpose of an emulator is to provide an accurate estimation of the robot cycle time for given robot setup and pick and place sequence. The emulator will obey the same control logic and employ the same timing delays as the robot motions will take. This requires precise characterization of the robot movement delay processes, initialization, and termination routines. Thus, the emulator is a quantitative definition of the robot system and its controller. It is a numerical offline line tool by which to obtain process rates for particular plates with respect to given setups and sequences.

This will provide precise estimates for production capacity analysis. The requirement is to be able to predict robot cycle time to within $\pm 5\%$ of the actual.

6.0 Exception Handling

6.1 Exception Detection and Recovery Requirements for the Controller

The robot controller is expected to have the intelligence to detect and handle exceptions as they occur. The nature of such exceptions and the recovery procedure will be determined at a later time.

6.2 Precision and Accuracy Requirements

Information pertaining to this will be supplied at a later date.

7.0 Data Supply, User Interfacing, and Program Management

A user-friendly interfacing system for operation of the robot and data management is needed. The BOM specification for each plate will be supplied in readable formatted files on a PC diskette. The data will contain part number names, placement location, and orientation, and any fudicial information for locating purposes.

Four primary functions are assumed, which are controlled via a task selection menu:

a. BOM specification tasks

   Here the operator will be able to supply a BOM and/or modify existing files.

b. Optimization tasks

   This function is divided into setup and sequencing of tasks:

   1) For setup, the operator may (i) choose own specification of a setup. This includes specification of a current setup or selection of a setup; (ii) operator may request for assistance in creation of a setup.

   2) For sequencing, the operator may supply the sequence or request the system to supply it. The optimized sequence should be accessible for viewing and editing by the operator.

c. Creation of the NC data, down loading, and task initiation.

d. This option will allow the operator to examine performance of the robot for a given setup and sequence for a particular plate.

In addition to the above, it is desirable for the PC system to collect statistics on online usage of the robot. This will include record of interruptions, exceptions occurred, number of runs, parts consumed, etc.

TABLE A1
30 mm Time-Distance

Location	Direction	Distance (mm)	Cycle Times (secs) T1	T2	AVERAGE	Total Distance Moved	Speed (mm/sec)
A	x	30.00	53	53	53.00	3000.00	56.60
	y	30.00	61	60	60.50	3000.00	49.59
	xy	42.43	68	68	68.00	4242.64	62.39
B	x	30.00	58	58	58.00	3000.00	51.72
	y	30.00	49	49	49.00	3000.00	61.22
	xy	42.43	59	59	59.00	4242.64	71.91
C	x	30.00	56	56	56.00	3000.00	53.57
	y	30.00	60	60	60.00	3000.00	50.00
	xy	42.43	55	55	55.00	4242.64	77.14
D	x	30.00	57	57	57.00	3000.00	52.63
	y	30.00	58	58	58.00	3000.00	51.72
	xy	42.43	67	67	67.00	4242.64	63.32
E	x	30.00	54	54	54.00	3000.00	55.56
	y	30.00	60	59	59.50	3000.00	50.42
	xy	42.43	61	61	61.00	4242.64	69.55
F	x	30.00	60	60	60.00	3000.00	50.00
	y	30.00	51	51	51.00	3000.00	58.82
	xy	42.43	61	61	61.00	4242.64	69.55
G	x	30.00	56	56	56.00	3000.00	53.57
	y	30.00	57	57	57.00	3000.00	52.63
	xy	42.43	63	63	63.00	4242.64	67.34

TABLE A2
100 mm Time-Distance

Location	Direction	Distance (mm)	Cycle Times (secs) T1	T2	AVERAGE	Total Distance Moved	Speed (mm/sec)
A	x	100.00	74	74	74.00	10000.00	135.14
	y	100.00	87	87	87.00	10000.00	114.94
	xy	141.42	103	103	103.00	14142.14	137.30
B	x	100.00	80	80	80.00	10000.00	125.00
	y	100.00	87	87	87.00	10000.00	114.94
	xy	141.42	86	86	86.00	14142.14	164.44
C	x	100.00	87	87	87.00	10000.00	114.94
	y	100.00	80	80	80.00	10000.00	125.00
	xy	141.42	87	87	87.00	14142.14	162.55
D	x	100.00	80	80	80.00	10000.00	125.00
	y	100.00	83	83	83.00	10000.00	120.48
	xy	141.42	101	101	101.00	14142.14	140.02
E	x	100.00	74	74	74.00	10000.00	135.14
	y	100.00	83	83	83.00	10000.00	120.48
	xy	141.42	85	85	85.00	14142.14	166.38
F	x	100.00	90	90	90.00	10000.00	111.11
	y	100.00	66	67	66.50	10000.00	150.38
	xy	141.42	85	85	85.00	14142.14	166.38
G	x	100.00	80	80	80.00	10000.00	125.00
	y	100.00	79	79	79.00	10000.00	126.58
	xy	141.42	92	92	92.00	14142.14	153.72

TABLE A3
250 mm Time-Distance

Location	Direction	Distance (mm)	Cycle Times (secs) T1	T2	AVERAGE	Total Distance Moved	Speed (mm/sec)
A	x	250.00	107	107	107.00	25000.00	233.64
	y	250.00	105	105	105.00	25000.00	238.10
	xy	353.55	110	111	110.50	35355.34	319.96
B	x	250.00	88	88	88.00	25000.00	284.09
	y	250.00	105	105	105.00	25000.00	238.10
	xy	353.55	114	114	114.00	35355.34	310.13
C	x	250.00	99	99	99.00	25000.00	252.53
	y	250.00	106	106	106.00	25000.00	235.85
	xy	353.55	116	116	116.00	35355.34	304.79
D	x	250.00	111	111	111.00	25000.00	225.23
	y	250.00	100	100	100.00	25000.00	250.00
	xy	353.55	107	106	106.50	35355.34	331.98
E	x	250.00	99	99	99.00	25000.00	252.53
	y	250.00	100	100	100.00	25000.00	250.00
	xy	353.55	120	119	119.50	35355.34	295.86
F	x	250.00	109	109	109.00	25000.00	229.36
	y	250.00	83	83	83.00	25000.00	301.20
	xy	353.55	118	118	118.00	35355.34	299.62
G	x	250.00	115	115	115.00	25000.00	217.39
	y	250.00	105	104	104.50	25000.00	239.23
	xy	353.55	102	102	102.00	35355.34	346.62

TABLE A4-
500 mm Time-Distance

Location	Direction	Distance (mm)	Cycle Times (secs) T1	T2	AVERAGE	Total Distance Moved	Speed (mm/sec)
A	x	500.00	119	118	118.50	50000.00	421.94
	y	500.00	0	0	0.00	50000.00	ERR
	xy	707.11	0	0	0.00	70710.68	ERR
B	x	500.00	121	122	121.50	50000.00	411.52
	y	500.00	0	0	0.00	50000.00	ERR
	xy	707.11	0	0	0.00	70710.68	ERR
C	x	500.00	124	125	124.50	50000.00	401.61
	y	500.00	140	140	140.00	50000.00	357.14
	xy	707.11	177	177	177.00	70710.68	399.50
D	x	500.00	136	136	136.00	50000.00	367.65
	y	500.00	122	121	121.50	50000.00	411.52
	xy	707.11	0	0	0.00	70710.68	ERR
E	x	500.00	140	139	139.50	50000.00	358.42
	y	500.00	129	129	129.00	50000.00	387.60
	xy	707.11	0	0	0.00	70710.68	ERR
F	x	500.00	144	145	144.50	50000.00	346.02
	y	500.00	132	131	131.50	50000.00	380.23
	xy	707.11	165	165	165.00	70710.68	428.55
G	x	500.00	0	0	0.00	50000.00	ERR
	y	500.00	138	137	137.50	50000.00	363.64
	xy	707.11	0	0	0.00	70710.68	ERR

Table A5: Theta 1 Joint Tests

Theta 1 (rads)	Moved from Home (1030,0) to: X(mm)	Y(mm)
0.01	1029.94	10.29
0.02	1029.79	20.59
0.03	1029.53	30.89
0.04	1029.17	41.18
0.05	1028.71	51.47
0.06	1028.14	61.76
0.07	1027.47	72.04
0.10	1024.85	102.83
0.30	983.99	304.38
0.40	948.69	401.10
0.50	903.90	493.80
0.60	850.00	581.58
0.70	787.79	663.54
0.79	728.32	728.32
0.80	717.60	738.88
1.00	556.51	866.71
1.05	515.00	892.00
1.10	467.20	917.94
1.20	373.22	960.00
1.30	275.52	992.46
1.40	175.06	1015.01
1.50	72.86	1027.42
1.57	0.00	1030.00
1.70	-132.71	1021.41
1.77	-204.62	1009.47
1.80	-234.02	1003.06
1.90	-332.98	974.69
2.00	-428.63	936.57
2.09	-515.00	892.00
2.10	-519.99	889.10
2.20	-606.15	832.75
2.30	-686.26	768.07
2.36	-728.32	728.32
2.40	-759.51	695.72
2.50	-825.18	616.42
2.60	-882.59	530.97
2.70	-931.19	440.20
2.80	-970.49	345.03
2.90	-1000.08	246.43
3.00	-1019.69	145.35
3.14	-1030.00	0.00
3.20	-1028.24	-60.12
3.30	-1017.10	-162.48
3.40	-995.80	-263.21
3.49	-967.88	-352.28

Table A6: Theta 2 Joint Tests

Theta 2 (rads)	Moved from Home (1030,0) (unless *) To: X(mm)	Y(mm)
0.01	1029.98	4.00
0.02	1029.92	8.00
0.04	1029.68	16.00
0.05	1029.50	20.00
0.06	1029.28	23.99
0.07	1029.02	27.98
0.10*	-39.93	1028.00
0.20*	-79.47	1022.03
0.30*	-118.21	1012.13
0.50*	-191.77	981.03
0.60*	-225.86	960.13
0.70*	-257.69	935.94
0.79	912.84	282.84
0.80*	-286.94	908.68
0.90*	-313.33	878.64
1.00*	-336.59	846.12
1.05	830.00	346.41
1.10	811.44	356.48
1.20	774.94	372.82
1.30	737.00	385.42
1.40	697.98	394.17
1.50	658.29	399.00
1.57	630.00	400.00
1.70	618.32	399.83
1.77	578.46	396.66
1.80	539.12	389.54
1.90	500.68	378.52
2.00	463.54	363.72
2.09	430.00	346.41
2.10	428.06	345.28
2.20	394.60	323.40
2.30	363.49	298.28
2.40	335.04	270.18
2.50	309.54	239.39
2.60	287.24	206.20
2.70	268.37	170.95
2.79	254.12	136.80

*These tests were moved from (0,1030)

Table A7: Theta 1 Average Speed

Theta 1 (rads)	Average Speed (rad/sec)
0.01	0.030
0.02	0.044
0.03	0.061
0.04	0.075
0.05	0.090
0.06	0.099
0.07	0.111
0.10	0.144
0.30	0.312
0.40	0.378
0.50	0.434
0.60	0.486
0.70	0.523
0.79	0.556
0.80	0.554
1.00	0.622
1.05	0.635
1.10	0.648
1.20	0.672
1.30	0.692
1.40	0.710
1.50	0.731
1.57	0.742
1.70	0.762
1.77	0.768
1.80	0.774
1.90	0.786
2.00	0.800
2.09	0.807
2.10	0.808
2.20	0.818
2.30	0.829
2.36	0.833
2.40	0.838
2.50	0.844
2.60	0.856
2.70	0.861
2.80	0.868
2.90	0.872
3.00	0.880
3.14	0.887
3.20	0.892
3.30	0.895
3.40	0.902
3.49	0.909

Table A8: Theta 2 Average Speed

Theta 2 (rads)	Average Speed (rad/sec)
0.01	0.031
0.02	0.050
0.04	0.084
0.05	0.099
0.06	0.114
0.07	0.128
0.10	0.166
0.20	0.275
0.30	0.364
0.50	0.518
0.60	0.586
0.70	0.646
0.79	0.693
0.80	0.700
0.90	0.750
1.00	0.795
1.05	0.814
1.10	0.833
1.20	0.872
1.30	0.906
1.40	0.937
1.50	0.967
1.57	0.986
1.70	0.994
1.77	1.018
1.80	1.041
1.90	1.064
2.00	1.084
2.09	1.101
2.10	1.105
2.20	1.122
2.30	1.140
2.40	1.156
2.50	1.171
2.60	1.184
2.70	1.199
2.79	1.210

ANALYSIS OF VARIANCE

$\theta_1 \leq 0.1$ radians

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	2	0.009663721	0.004831861	2051.136	0.0001
ERROR	5	0.000011779	0.0000023557		
C TOTAL	7	0.009675500			

ROOT MSE	0.001534829	R-SQUARE	0.9988
DEP MEAN	0.08175	ADJ R-SQ	0.9983
C.V.	1.877467		

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR H0: PARAMETER=0	PROB > |T|
INTERCEP	1	0.01342624	0.001823083	7.365	0.0007
RADIANS	1	1.67468590	0.07650899	21.889	0.0001
R	1	-3.74127501	0.68255284	-5.481	0.0028

KEY TO THE ANOVA:

The dependent variable = speed

intercept

radians = radians moved

R = (radians moved)^2

ANALYSIS OF VARIANCE

$0.1 < \theta_1 \leq 1.0$ radians

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	2	0.16276260	0.08138130	2334.828	0.0001
ERROR	5	0.000174277	0.000034855		
C TOTAL	7	0.16293688			

ROOT MSE	0.005903844	R-SQUARE	0.9989
DEP MEAN	0.431875	ADJ R-SQ	0.9985
C.V.	1.367026		

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR H0: PARAMETER=0	PROB > |T|
INTERCEP	1	0.05884261	0.007751552	7.591	0.0006
RADIANS	1	0.93872025	0.03029028	30.991	0.0001
R	1	-0.38035738	0.02667426	-14.259	0.0001

KEY TO THE ANOVA:

The dependent variable = speed

intercept

radians = radians moved

R = (radians moved)^2

ANALYSIS OF VARIANCE

$1.0 < \theta_1 \leq 2.0$ radians

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	2	0.04349475	0.02174738	8261.543	0.0001
ERROR	10	0.000026324	.00000263236		
C TOTAL	12	0.04352108			

ROOT MSE	0.001622456	R-SQUARE	0.9994
DEP MEAN	0.7186154	ADJ R-SQ	0.9993
C.V.	0.2257753		

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR H0: PARAMETER=0	PROB > |T|
INTERCEP	1	0.29771285	0.01112653	26.757	0.0001
RADIANS	1	0.40144282	0.01555801	25.803	0.0001
R	1	-0.07573686	0.005230860	-14.479	0.0001

KEY TO THE ANOVA:

The dependent variable = speed

intercept

radians = radians moved

R = (radians moved)^2

ANALYSIS OF VARIANCE

$2.0 < \theta_1 \leq 3.0$ radians

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	2	0.008470369	0.004235185	2245.404	0.0001
ERROR	10	0.000018862	.00000188616		
C TOTAL	12	0.008489231			

ROOT MSE	0.001373374	R-SQUARE	0.9978
DEP MEAN	0.8395385	ADJ R-SQ	0.9973
C.V.	0.1635868		

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR H0: PARAMETER=0	PROB > |T|
INTERCEP	1	0.47231027	0.02721838	17.353	0.0001
RADIANS	1	0.21760598	0.02208164	9.855	0.0001
R	1	-0.02729981	0.004421441	-6.174	0.0001

KEY TO THE ANOVA:

The dependent variable = speed

intercept

radians = radians moved

R = (radians moved)^2

ANALYSIS OF VARIANCE

$3.0 < \theta_1 \leq 3.49$ radians

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	2	0.000534317	0.000267159	177.479	0.0008
ERROR	3	.00000451589	0.0000015053		
C TOTAL	5	0.000538833			

ROOT MSE	0.001226906	R-SQUARE	0.9916
DEP MEAN	0.8941667	ADJ R-SQ	0.9860
C.V.	0.1372122		

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR H0: PARAMETER=0	PROB > |T|
INTERCEP	1	0.95872937	0.21436003	4.473	0.0208
RADIANS	1	-0.09814166	0.13220993	-0.742	0.5117
R	1	0.02399714	0.02034540	1.179	0.3232

KEY TO THE ANOVA:

The dependent variable = speed

intercept

radians = radians moved

R = (radians moved)^2

ANALYSIS OF VARIANCE

$\theta_2 \leq 0.1$ radians

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	2	0.01274119	0.006370597	31647.484	0.0001
ERROR	4	8.05195E-07	2.01299E-07		
C TOTAL	6	0.01274200			

ROOT MSE	0.0004486632	R-SQUARE	0.9999
DEP MEAN	0.096	ADJ R-SQ	0.9999
C.V.	0.4673575		

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR H0: PARAMETER=0	PROB > |T|
INTERCEP	1	0.01246753	0.000540178	23.080	0.0001
RADIANS	1	1.93636364	0.02237706	86.533	0.0001
R	1	-4.02597403	0.20054805	-20.075	0.0001

KEY TO THE ANOVA:

The dependent variable = speed

intercept

radians = radians moved

R = (radians moved)^2

ANALYSIS OF VARIANCE

$0.1 < \Theta_2 \leq 1.0$ radians

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	2	0.41205193	0.20602596	11078.954	0.0001
ERROR	7	0.000130173	0.000018596		
C TOTAL	9	0.41218210			

ROOT MSE	0.004312326	R-SQUARE	0.9997
DEP MEAN	0.5493	ADJ R-SQ	0.9996
C.V.	0.7850585		

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR H0: PARAMETER=0	PROB > |T|
INTERCEP	1	0.06901168	0.005029891	13.720	0.0001
RADIANS	1	1.07254204	0.02145631	49.987	0.0001
R	1	-0.35042914	0.01938106	-18.081	0.0001

KEY TO THE ANOVA:

The dependent variable = speed

intercept

radians = radians moved

R = (radians moved)^2

ANALYSIS OF VARIANCE

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	2	0.11135602	0.05567801	55767.535	0.0001
ERROR	10	.00000998395	9.98395E-07		
C TOTAL	12	0.11136600			

ROOT MSE	0.000999197	R-SQUARE	0.9999
DEP MEAN	0.947	ADJ R-SQ	0.9999
C.V.	0.1055118		

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR H0: PARAMETER=0	PROB > |T|
INTERCEP	1	0.29284669	0.006632272	44.155	0.0001
RADIANS	1	0.61072457	0.009246064	66.052	0.0001
R	1	-0.10786340	0.003110146	-34.681	0.0001

KEY TO THE ANOVA:

The dependent variable = speed

intercept

radians = radians moved

R = (radians moved)^2

ANALYSIS OF VARIANCE

$2.0 < \theta_2 \leq 2.792$ radians

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	2	0.01717512	0.008587561	13162.699	0.0001
ERROR	7	.00000456691	6.52416E-07		
C TOTAL	9	0.01717969			

ROOT MSE	0.000807723	R-SQUARE	0.9997
DEP MEAN	1.14719	ADJ R-SQ	0.9997
C.V.	0.07040882		

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR H0: PARAMETER=0	PROB > |T|
INTERCEP	1	0.47334158	0.02633577	17.973	0.0001
RADIANS	1	0.41107989	0.02222582	18.496	0.0001
R	1	-0.05279303	0.004642414	-11.372	0.0001

KEY TO THE ANOVA:

The dependent variable = speed

intercept

radians = radians moved

R = (radians moved)^2

Table A11: Z-axis Tests
(no weight)

Z-axis (mm)	Average Time (secs)
- 10	10.48
- 20	12.46
- 40	15.41
- 60	17.53
- 80	19.36
-100	21.26
-120	23.00
-140	24.65
-160	26.22
-180	27.74
-200	29.42
-220	31.50
-240	33.27
-250	34.20

z axis tests of 9.5 cycles

Table A12: Z-axis tests for 7547 Application

Z-AXIS MOVES	TIME REQUIRED
0 to clear	.8045 sec
clear to plate (grasp) to clear	1.663 sec
clear to plate (release) to clear	1.658 sec
clear to magazine (grasp) to clear	2.2 sec
clear to magazine (release) to clear	2.043 sec

Z-axis values

clear = -40 mm
plate = -53 mm
magazine = -73 mm

These times were developed based on the mock-up setup. These were constant values added to the cycle time.

Table A13: Magazine Status Verified

Magazines Status' during five cycles of NBOR5

Cycle #	Cartridge #	Magazine 1	Magazine 2
1	1	5	6
	2	4	6
	3	3	6
	4	3	6
	5	4	6
	6	6	6
	7	3	6
	8	3	6
	9	2	6
	10	5	6
	11	1	6
	12	6	6
	13	4	6
	14	3	6
2	1	4	6
	2	2	6
	3	0	6
	4	0	6
	5	2	6
	6	4	6
	7	0	6
	8	0	6
	9	0	6
	10	4	6
	11	0	6
	12	2	6
	13	2	6
	14	0	6
3	1	4	5
	2	2	4
	3	6	3
	4	6	3
	5	2	3
	6	4	4
	7	6	2
	8	6	4
	9	6	4
	10	4	4
	11	6	5
	12	2	2
	13	2	5
	14	6	4

Table A13 continued

Cycle #	Cartridge #	Magazine 1	Magazine 2
4	1	4	4
	2	2	2
	3	6	0
	4	6	0
	5	2	0
	6	4	2
	7	6	0
	8	6	0
	9	6	2
	10	4	2
	11	6	4
	12	2	0
	13	2	2
	14	6	2
5	1	3	4
	2	0	2
	3	3	6
	4	3	6
	5	0	6
	6	4	2
	7	3	6
	8	3	6
	9	2	2
	10	3	2
	11	1	4
	12	2	6
	13	0	2
	14	3	2

** When switching to alternate magazine the cartridges with 0 dispensers in the current magazine are automatically filled.

Table A14: Theta 1
Regression Equations Verified
(medium speed, no weight)

Theta 1 (rads)	Average Time (sec)	Predict Time (sec)	Percent Error
0.01	33.5	33.58	0.24
0.02	45.2	44.05	-2.54
0.03	49.6	49.76	0.32
0.04	53.2	53.75	1.03
0.05	55.8	56.94	2.04
0.06	60.8	59.74	-1.74
0.07	63.0	62.32	-1.08
0.10	69.6	69.69	0.13
0.30	96.1	97.98	1.96
0.40	105.8	107.12	1.25
0.50	115.2	115.46	0.23
0.60	123.6	123.69	0.07
0.70	133.8	132.20	-1.20
0.79	141.2	140.33	-0.62
0.80	144.5	141.27	-2.24
1.00	160.8	162.05	0.78
1.05	164.9	165.17	0.16
1.10	169.7	169.85	0.09
1.20	178.6	179.00	0.22
1.30	187.8	187.97	0.09
1.40	197.3	196.83	-0.24
1.50	205.2	205.63	0.21
1.57	211.5	211.79	0.14
1.70	223.2	223.30	0.04
1.77	230.5	229.57	-0.40
1.80	232.5	232.27	-0.10
1.90	241.8	241.40	-0.17
2.00	250.0	250.72	0.29
2.09	258.9	258.72	-0.07
2.10	259.9	259.62	-0.11
2.20	269.1	268.66	-0.16
2.30	277.6	277.66	0.02
2.36	282.9	283.05	0.05
2.40	286.4	286.64	0.08
2.50	296.1	295.62	-0.16
2.60	303.9	304.62	0.24
2.70	313.5	313.66	0.05
2.80	322.5	322.75	0.08
2.90	332.5	331.90	-0.18
3.00	340.7	340.76	0.02
3.14	353.9	354.03	0.04
3.20	358.8	359.33	0.15
3.30	368.6	368.17	-0.12

Table A14: Theta 1 continued

Theta 1 (rads)	Average Time (sec)	Predict Time (sec)	Percent Error
3.40	377.1	376.69	-0.11
3.49	383.95	384.14	0.05

Tests consisted of 50 cycles.

Table A15: Theta 2
Regression Equations Verified
(medium speed, no weight)

Theta 2 (rads)	Average Time (sec)	Predict Time (sec)	Percent Error
0.01	32.2	31.79	-1.27
0.02	39.8	40.31	1.28
0.04	47.4	47.90	1.05
0.05	50.4	50.38	-0.04
0.06	52.8	52.55	-0.47
0.07	54.9	54.56	-0.62
0.10	60.1	60.30	0.33
0.20	72.8	74.24	1.98
0.30	82.4	83.55	1.40
0.50	96.5	96.64	0.15
0.60	102.4	102.38	-0.02
0.70	108.4	108.07	-0.30
0.79	113.2	113.31	0.10
0.80	114.3	113.90	-0.35
0.90	120.0	120.00	0.00
1.00	125.8	126.49	0.55
1.05	128.6	128.82	0.17
1.10	132.0	131.89	-0.08
1.20	137.7	137.89	0.14
1.30	143.6	143.75	0.10
1.40	149.4	149.52	0.08
1.50	155.1	155.27	0.11
1.57	159.2	159.29	0.06
1.70	161.0	161.02	0.01
1.77	167.0	166.80	-0.12
1.80	172.9	172.67	-0.13
1.90	178.6	178.64	0.02
2.00	184.5	184.74	0.13
2.09	189.9	189.68	-0.12
2.10	190.1	190.25	0.08
2.20	196.0	196.05	0.03
2.30	201.7	201.84	0.07
2.40	207.6	207.65	0.02
2.50	213.5	213.48	-0.01
2.60	219.6	219.37	-0.10
2.70	225.2	225.31	0.05
2.79	230.8	230.72	-0.03

Tests consisted of 50 cycles.

```
PIC: STATIC COUNTER;

CTR1 : STATIC COUNTER;  CTR2:STATIC COUNTER;
CTR3 : STATIC COUNTER;  CTR4:STATIC COUNTER;
CTR5 : STATIC COUNTER;  CTR6:STATIC COUNTER;
CTR7 : STATIC COUNTER;  CTR8:STATIC COUNTER;
CTR9 : STATIC COUNTER;  CTR10:STATIC COUNTER;
CTR11: STATIC COUNTER;  CTR12:STATIC COUNTER;
CTR13: STATIC COUNTER;  CTR14:STATIC COUNTER;

CAT1 : STATIC COUNTER; CAT2 : STATIC COUNTER;
CAT3 : STATIC COUNTER; CAT4 : STATIC COUNTER;
CAT5 : STATIC COUNTER; CAT6 : STATIC COUNTER;
CAT7 : STATIC COUNTER; CAT8 : STATIC COUNTER;
CAT9 : STATIC COUNTER; CAT10: STATIC COUNTER;
CAT11: STATIC COUNTER; CAT12: STATIC COUNTER;
CAT13: STATIC COUNTER; CAT14: STATIC COUNTER;

C1  : STATIC COUNTER;   C2:STATIC COUNTER;
C3  : STATIC COUNTER;   C4:STATIC COUNTER;
C5  : STATIC COUNTER;   C6:STATIC COUNTER;
C7  : STATIC COUNTER;   C8:STATIC COUNTER;
C9  : STATIC COUNTER;   C10:STATIC COUNTER;
C11 : STATIC COUNTER;   C12:STATIC COUNTER;
C13 : STATIC COUNTER;   C14:STATIC COUNTER;

MAG_NO : STATIC COUNTER;

MAG_1 : SUBR;
                  --------NBOR5---------
X : NEW <-752,-714,-676,-638,-600,-562,-562,-600,-600,-600,-562,-562,
          -600,-638,-638,-638,-638,-638,-676,-676,-676,-676,-676,-714,
         -714,-714,-714,-714,-752,-752,-752,-752>;
Y : NEW <473,473,473,473,473,511,549,549,587,625,625,663,663,663,625,
        587,549,511,511,549,587,625,663,663,625,587,549,511,549,587,625,
        663>;
DES : NEW <3,8,9,2,13,14,3,4,1,11,10,2,11,4,13,3,11,14,9,14,9,11,9,
           11,5,4,8,8,5,7,7,7>;
DES2: NEW <3,12,12,2,14,5,3,4,1,7,9,2,7,4,6,3,13,5,12,5,10,7,8,7,6,4,
           11,12,14,10,9,8>;
--          ERROR1: SUBR;
--                PICK:NEW -52;CLEAR:NEW -40.0;PLATPROB:NEW 1;
--                SETC(PIC,0);
--                ONCE:TESTI(16,0,TRY);BRANCH(QUIT);TRY:ZMOVE(PICK);
--                GRASP;ZMOVE(CLEAR);TESTC(PIC,3,ALARM);INCR(PIC);
--                BRANCH(ONCE);ALARM:WRITEO(PLATPROB,1);BREAKPOINT;
--                WRITEO(PLATPROB,0);QUIT:
--                END;
            PICKUP : SUBR(A,B);           --SUBR TO PICKUP PART ONLY
               PICK  : NEW -52;   CLEAR : NEW -40.0;
                  PMOVE(PT(A,B,CLEAR,0));
                  ZMOVE(PICK);  DELAY(0.5);GRASP;
                  ZMOVE(CLEAR);
                  END;
            PUTDOWN : SUBR(A,B);          --PUTS DOWN PART ON MAGAZINE
```

```
                DROPOFF   : NEW -71;  CLEAR : NEW -40.0;
              PMOVE(PT(A,B,CLEAR,0));
              ZMOVE(DROPOFF);  DELAY(0.5); RELEASE;
              ZMOVE(CLEAR);
              END;
        PUTDOWN2: SUBR(A,B);
              DROPOFF : NEW -71;
              CLEAR : NEW -40;
              PMOVE(PT(B,A,CLEAR,0)); ZMOVE(DROPOFF);
              DELAY(0.5); RELEASE;ZMOVE(CLEAR);
              END;
        PUTPLATE: SUBR(A,B);
              PICK  : NEW -52; CLEAR : NEW -40.0;
              PMOVE(PT(A,B,CLEAR,90));
              ZMOVE(PICK); DELAY(0.5); RELEASE;
              ZMOVE(CLEAR);
              END;
        MVDLTA: SUBR;
              DROPOFF : NEW -71; CLEAR : NEW -40.0;
              DPMOVE(<0,35,0,0>);
              ZMOVE(DROPOFF);  DELAY(0.5); GRASP;
              ZMOVE(CLEAR);
              END;
        MVDLTA2: SUBR;
              DROPOFF : NEW -71; CLEAR : NEW -40.0;
              DPMOVE(<-35,0,0,0>);
              ZMOVE(DROPOFF);  DELAY(0.5); GRASP;
              ZMOVE(CLEAR);
              END;
 MAGZ1 : SUBR(DESTN);                          --BRANCH TO LABEL

Y_LC1  : NEW   -938;   Y_LC2 : NEW -905;
Y_LC3  : NEW   -872;   Y_LC4 : NEW -839;
Y_LC5  : NEW   -806;   Y_LC6 : NEW -773;
Y_LC7  : NEW   -740;   Y_LC8 : NEW -707;
Y_LC9  : NEW   -674;   Y_LC10 : NEW -641;
Y_LC11 : NEW   -608;   Y_LC12 : NEW -575;
Y_LC13 : NEW   -542;   Y_LC14 : NEW -509;

Y_LOC1 : NEW    205;    Y_LOC2 : NEW 240;
Y_LOC3 : NEW    275;    Y_LOC4 : NEW 310;
Y_LOC5 : NEW    345;    Y_LOC6 : NEW 380;
  SETC(C1,1);  SETC(C2,2);  SETC(C3,3);
  SETC(C4,4);  SETC(C5,5);  SETC(C6,6);
  SETC(C7,7);  SETC(C8,8);  SETC(C9,9);
  SETC(C10,10);  SETC(C11,11);  SETC(C12,12);
  SETC(C13,13);  SETC(C14,14);

TESTC(C1,DESTN, CART_1);      TESTC(C2, DESTN, CART_2);
TESTC(C3,DESTN, CART_3);      TESTC(C4, DESTN, CART_4);
TESTC(C5,DESTN, CART_5);      TESTC(C6, DESTN, CART_6);
TESTC(C7,DESTN, CART_7);      TESTC(C8, DESTN, CART_8);
TESTC(C9,DESTN, CART_9);      TESTC(C10, DESTN, CART_10);
TESTC(C11,DESTN, CART_11);    TESTC(C12, DESTN, CART_12);
TESTC(C13,DESTN, CART_13);    TESTC(C14, DESTN, CART_14);
```

```
CART_1:
            TESTC(CTR1, 6, EXCEP1); TESTC(CTR1, 5, MV16);
            TESTC(CTR1, 4, MV15);   TESTC(CTR1, 3, MV14);
            TESTC(CTR1, 2, MV13);   TESTC(CTR1, 1, MV12);
            TESTC(CTR1, 0, MV11);
      MV11: PUTDOWN(Y_LC1,Y_LOC1);
            INCR(CTR1); MVDLTA; BRANCH(QUIT);
      MV12: PUTDOWN(Y_LC1,Y_LOC2);
            INCR(CTR1); MVDLTA; BRANCH(QUIT);
      MV13: PUTDOWN(Y_LC1,Y_LOC3);
            INCR(CTR1); MVDLTA; BRANCH(QUIT);
      MV14: PUTDOWN(Y_LC1,Y_LOC4);
            INCR(CTR1); MVDLTA; BRANCH(QUIT);
      MV15: PUTDOWN(Y_LC1,Y_LOC5);
            INCR(CTR1); MVDLTA; BRANCH(QUIT);
      MV16: PUTDOWN(Y_LC1,Y_LOC6);
            INCR(CTR1); MVDLTA;BRANCH(QUIT);
      EXCEP1:                         --MOVE TO NEXT MAGAZINE
            SETC(CTR1,0);
            BREAKPOINT;
            BRANCH(MAG_2);
CART_2:     TESTC(CTR2, 6, EXCEP2); TESTC(CTR2, 5, MV26);
            TESTC(CTR2, 4, MV25);   TESTC(CTR2, 3, MV24);
            TESTC(CTR2, 2, MV23);   TESTC(CTR2, 1, MV22);
            TESTC(CTR2, 0, MV21);
      MV21: PUTDOWN(Y_LC2, Y_LOC1);
            INCR(CTR2); MVDLTA;BRANCH(QUIT);
      MV22: PUTDOWN(Y_LC2,Y_LOC2);
            INCR(CTR2); MVDLTA;BRANCH(QUIT);
      MV23: PUTDOWN(Y_LC2,Y_LOC3);
            INCR(CTR2); MVDLTA;BRANCH(QUIT);
      MV24: PUTDOWN(Y_LC2, Y_LOC4);
            INCR(CTR2); MVDLTA;BRANCH(QUIT);
      MV25: PUTDOWN(Y_LC2, Y_LOC5);
            INCR(CTR2); MVDLTA;BRANCH(QUIT);
      MV26: PUTDOWN(Y_LC2,Y_LOC6);
            INCR(CTR2); MVDLTA;BRANCH(QUIT);
EXCEP2:     SETC(CTR2,0);
            BREAKPOINT;
            BRANCH(MAG_2);
CART_3:     TESTC(CTR3, 6, EXCEP3); TESTC(CTR3, 5, MV36);
            TESTC(CTR3, 4, MV35);   TESTC(CTR3, 3, MV34);
            TESTC(CTR3, 2, MV33);   TESTC(CTR3, 1, MV32);
            TESTC(CTR3, 0, MV31);
      MV31: PUTDOWN(Y_LC3,Y_LOC1);
            INCR(CTR3); MVDLTA;BRANCH(QUIT);
      MV32: PUTDOWN(Y_LC3,Y_LOC2);
            INCR(CTR3); MVDLTA;BRANCH(QUIT);
      MV33: PUTDOWN(Y_LC3,Y_LOC3);
            INCR(CTR3); MVDLTA;BRANCH(QUIT);
      MV34: PUTDOWN(Y_LC3,Y_LOC4);
            INCR(CTR3); MVDLTA;BRANCH(QUIT);
      MV35: PUTDOWN(Y_LC3,Y_LOC5);
            INCR(CTR3); MVDLTA;BRANCH(QUIT);
      MV36: PUTDOWN(Y_LC3, Y_LOC6);
```

```
            INCR(CTR3); MVDLTA;BRANCH(QUIT);
EXCEP3:     SETC(CTR3,0);
            BREAKPOINT;
            BRANCH(MAG_2);
CART_4:     TESTC(CTR4, 6, EXCEP4); TESTC(CTR4, 5, MV46);
            TESTC(CTR4, 4, MV45);   TESTC(CTR4, 3, MV44);
            TESTC(CTR4, 2, MV43);   TESTC(CTR4, 1, MV42);
            TESTC(CTR4, 0, MV41);
      MV41: PUTDOWN(Y_LC4,Y_LOC1);
            INCR(CTR4); MVDLTA;BRANCH(QUIT);
      MV42: PUTDOWN(Y_LC4,Y_LOC2);
            INCR(CTR4); MVDLTA;BRANCH(QUIT);
      MV43: PUTDOWN(Y_LC4,Y_LOC3);
            INCR(CTR4); MVDLTA;BRANCH(QUIT);
      MV44: PUTDOWN(Y_LC4,Y_LOC4);
            INCR(CTR4); MVDLTA;BRANCH(QUIT);
      MV45: PUTDOWN(Y_LC4,Y_LOC5);
            INCR(CTR4); MVDLTA;BRANCH(QUIT);
      MV46: PUTDOWN(Y_LC4,Y_LOC6);
            INCR(CTR4); MVDLTA;BRANCH(QUIT);
EXCEP4:     SETC(CTR4,0);
            BREAKPOINT;
            BRANCH(MAG_2);
CART_5:     TESTC(CTR5, 6, EXCEP5); TESTC(CTR5, 5, MV56);
            TESTC(CTR5, 4, MV55);   TESTC(CTR5, 3, MV54);
            TESTC(CTR5, 2, MV53);   TESTC(CTR5, 1, MV52);
            TESTC(CTR5, 0, MV51);
      MV51: PUTDOWN(Y_LC5,Y_LOC1);
            INCR(CTR5); MVDLTA;BRANCH(QUIT);
      MV52: PUTDOWN(Y_LC5,Y_LOC2);
            INCR(CTR5); MVDLTA;BRANCH(QUIT);
      MV53: PUTDOWN(Y_LC5,Y_LOC3);
            INCR(CTR5); MVDLTA;BRANCH(QUIT);
      MV54: PUTDOWN(Y_LC5,Y_LOC4);
            INCR(CTR5); MVDLTA;BRANCH(QUIT);
      MV55: PUTDOWN(Y_LC5,Y_LOC5);
            INCR(CTR5); MVDLTA;BRANCH(QUIT);
      MV56: PUTDOWN(Y_LC5,Y_LOC6);
            INCR(CTR5); MVDLTA;BRANCH(QUIT);
EXCEP5:     TESTC(CTR13,6,NEXT5);          --TEST IF CART 13
            BRANCH(CART_13);               --IS FULL
NEXT5:      SETC(CTR5,0);                  --MOVE TO NEXT
            SETC(CTR13,0);                 --MAGAZINE
            BRANCH(MAG_2);
CART_6:     TESTC(CTR6, 6, EXCEP6); TESTC(CTR6, 5, MV66);
            TESTC(CTR6, 4, MV65);   TESTC(CTR6, 3, MV64);
            TESTC(CTR6, 2, MV63);   TESTC(CTR6, 1, MV62);
            TESTC(CTR6, 0, MV61);
      MV61: PUTDOWN(Y_LC6,Y_LOC1);
            INCR(CTR6); MVDLTA;BRANCH(QUIT);
      MV62: PUTDOWN(Y_LC6,Y_LOC2);
            INCR(CTR6); MVDLTA;BRANCH(QUIT);
      MV63: PUTDOWN(Y_LC6,Y_LOC3);
            INCR(CTR6); MVDLTA;BRANCH(QUIT);
      MV64: PUTDOWN(Y_LC6,Y_LOC4);
```

```
                INCR(CTR6); MVDLTA;BRANCH(QUIT);
        MV65:   PUTDOWN(Y_LC6,Y_LOC5);
                INCR(CTR6); MVDLTA;BRANCH(QUIT);
        MV66:   PUTDOWN(Y_LC6,Y_LOC6);
                INCR(CTR6); MVDLTA;BRANCH(QUIT);
EXCEP6:         TESTC(CTR6,6,NEXT66);
                BRANCH(CART_6);
NEXT66:         TESTC(CTR7,6,NEXT67);
                BRANCH(CART_7);
NEXT67:         TESTC(CTR8,6,NEXT68);
                BRANCH(CART_8);
NEXT68:         TESTC(CTR9,6,NEXT69);
                BRANCH(CART_9);
NEXT69:         TESTC(CTR10,6,NEXTMAG);
                BRANCH(CART_10);
NEXTMAG:        SETC(CTR6,0); SETC(CTR7,0); SETC(CTR8,0);
                SETC(CTR9,0); SETC(CTR10,0);
                BRANCH(MAG_2);
CART_7:         TESTC(CTR7, 6, EXCEP6); TESTC(CTR7, 5, MV76);
                TESTC(CTR7, 4, MV75);   TESTC(CTR7, 3, MV74);
                TESTC(CTR7, 2, MV73);   TESTC(CTR7, 1, MV72);
                TESTC(CTR7, 0, MV71);
        MV71:   PUTDOWN(Y_LC7,Y_LOC1);
                INCR(CTR7); MVDLTA;BRANCH(QUIT);
        MV72:   PUTDOWN(Y_LC7,Y_LOC2);
                INCR(CTR7); MVDLTA;BRANCH(QUIT);
        MV73:   PUTDOWN(Y_LC7,Y_LOC3);
                INCR(CTR7); MVDLTA;BRANCH(QUIT);
        MV74:   PUTDOWN(Y_LC7,Y_LOC4);
                INCR(CTR7); MVDLTA;BRANCH(QUIT);
        MV75:   PUTDOWN(Y_LC7,Y_LOC5);
                INCR(CTR7); MVDLTA;BRANCH(QUIT);
        MV76:   PUTDOWN(Y_LC7,Y_LOC6);
                INCR(CTR7); MVDLTA;BRANCH(QUIT);
CART_8:         TESTC(CTR8, 6, EXCEP6); TESTC(CTR8, 5, MV86);
                TESTC(CTR8, 4, MV85);   TESTC(CTR8, 3, MV84);
                TESTC(CTR8, 2, MV83);   TESTC(CTR8, 1, MV82);
                TESTC(CTR8, 0, MV81);
        MV81:   PUTDOWN(Y_LC8,Y_LOC1);
                INCR(CTR8); MVDLTA;BRANCH(QUIT);
        MV82:   PUTDOWN(Y_LC8,Y_LOC2);
                INCR(CTR8); MVDLTA;BRANCH(QUIT);
        MV83:   PUTDOWN(Y_LC8,Y_LOC3);
                INCR(CTR8); MVDLTA;BRANCH(QUIT);
        MV84:   PUTDOWN(Y_LC8,Y_LOC4);
                INCR(CTR8); MVDLTA;BRANCH(QUIT);
        MV85:   PUTDOWN(Y_LC8,Y_LOC5);
                INCR(CTR8); MVDLTA;BRANCH(QUIT);
        MV86:   PUTDOWN(Y_LC8,Y_LOC6);
                INCR(CTR8); MVDLTA;BRANCH(QUIT);
CART_9:         TESTC(CTR9, 6, EXCEP6); TESTC(CTR9, 5, MV96);
                TESTC(CTR9, 4, MV95);   TESTC(CTR9, 3, MV94);
                TESTC(CTR9, 2, MV93);   TESTC(CTR9, 1, MV92);
                TESTC(CTR9, 0, MV91);
        MV91:   PUTDOWN(Y_LC9,Y_LOC1);
```

```
            INCR(CTR9); MVDLTA;BRANCH(QUIT);
      MV92: PUTDOWN(Y_LC9,Y_LOC2);
            INCR(CTR9); MVDLTA;BRANCH(QUIT);
      MV93: PUTDOWN(Y_LC9,Y_LOC3);
            INCR(CTR9); MVDLTA;BRANCH(QUIT);
      MV94: PUTDOWN(Y_LC9,Y_LOC4);
            INCR(CTR9); MVDLTA;BRANCH(QUIT);
      MV95: PUTDOWN(Y_LC9,Y_LOC5);
            INCR(CTR9); MVDLTA;BRANCH(QUIT);
      MV96: PUTDOWN(Y_LC9,Y_LOC6);
            INCR(CTR9); MVDLTA;BRANCH(QUIT);
CART_10:    TESTC(CTR10, 6, EXCEP6);TESTC(CTR10, 5, MV106);
            TESTC(CTR10, 4, MV105); TESTC(CTR10, 3, MV104);
            TESTC(CTR10, 2, MV103); TESTC(CTR10, 1, MV102);
            TESTC(CTR10, 0, MV101);
     MV101:PUTDOWN(Y_LC10,Y_LOC1);
            INCR(CTR10); MVDLTA;BRANCH(QUIT);
     MV102:PUTDOWN(Y_LC10,Y_LOC2);
            INCR(CTR10); MVDLTA;BRANCH(QUIT);
     MV103:PUTDOWN(Y_LC10,Y_LOC3);
            INCR(CTR10); MVDLTA;BRANCH(QUIT);
     MV104:PUTDOWN(Y_LC10,Y_LOC4);
            INCR(CTR10); MVDLTA;BRANCH(QUIT);
     MV105:PUTDOWN(Y_LC10,Y_LOC5);
            INCR(CTR10); MVDLTA;BRANCH(QUIT);
     MV106:PUTDOWN(Y_LC10,Y_LOC6);
            INCR(CTR10); MVDLTA;BRANCH(QUIT);
CART_11:    TESTC(CTR11, 6, EXCEP11); TESTC(CTR11, 5, MV116);
            TESTC(CTR11, 4, MV115); TESTC(CTR11, 3, MV114);
            TESTC(CTR11, 2, MV113); TESTC(CTR11, 1, MV112);
            TESTC(CTR11, 0, MV111);
     MV111:PUTDOWN(Y_LC11,Y_LOC1);
            INCR(CTR11); MVDLTA;BRANCH(QUIT);
     MV112:PUTDOWN(Y_LC11,Y_LOC2);
            INCR(CTR11); MVDLTA;BRANCH(QUIT);
     MV113:PUTDOWN(Y_LC11,Y_LOC3);
            INCR(CTR11); MVDLTA;BRANCH(QUIT);
     MV114:PUTDOWN(Y_LC11,Y_LOC4);
            INCR(CTR11); MVDLTA;BRANCH(QUIT);
     MV115:PUTDOWN(Y_LC11,Y_LOC5);
            INCR(CTR11); MVDLTA;BRANCH(QUIT);
     MV116:PUTDOWN(Y_LC11,Y_LOC6);
            INCR(CTR11); MVDLTA;BRANCH(QUIT);
EXCEP11:    TESTC(CTR11,6,NEXT11);
            BRANCH(CART_11);
NEXT11:     TESTC(CTR12,6,NEXT12);
            BRANCH(CART_12);
NEXT12:     SETC(CTR11,0);
            SETC(CTR12,0);
            BRANCH(MAG_2);
CART_12:    TESTC(CTR12, 6, EXCEP11); TESTC(CTR12, 5, MV126);
            TESTC(CTR12, 4, MV125); TESTC(CTR12, 3, MV124);
            TESTC(CTR12, 2, MV123); TESTC(CTR12, 1, MV122);
            TESTC(CTR12, 0, MV121);
     MV121:PUTDOWN(Y_LC12,Y_LOC1);
```

```
                INCR(CTR12); MVDLTA;BRANCH(QUIT);
          MV122:PUTDOWN(Y_LC12,Y_LOC2);
                INCR(CTR12); MVDLTA;BRANCH(QUIT);
          MV123:PUTDOWN(Y_LC12,Y_LOC3);
                INCR(CTR12); MVDLTA;BRANCH(QUIT);
          MV124:PUTDOWN(Y_LC12,Y_LOC4);
                INCR(CTR12); MVDLTA;BRANCH(QUIT);
          MV125:PUTDOWN(Y_LC12,Y_LOC5);
                INCR(CTR12); MVDLTA;BRANCH(QUIT);
          MV126:PUTDOWN(Y_LC12,Y_LOC6);
                INCR(CTR12); MVDLTA;BRANCH(QUIT);
                BRANCH(CART_11);
    CART_13:    TESTC(CTR13, 6, EXCEP5); TESTC(CTR12, 5, MV136);
                TESTC(CTR13, 4, MV135); TESTC(CTR13, 3, MV134);
                TESTC(CTR13, 2, MV133); TESTC(CTR13, 1, MV132);
                TESTC(CTR13, 0, MV131);
          MV131:PUTDOWN(Y_LC13,Y_LOC1);
                INCR(CTR13); MVDLTA;BRANCH(QUIT);
          MV132:PUTDOWN(Y_LC13,Y_LOC2);
                INCR(CTR13); MVDLTA;BRANCH(QUIT);
          MV133:PUTDOWN(Y_LC13,Y_LOC3);
                INCR(CTR13); MVDLTA;BRANCH(QUIT);
          MV134:PUTDOWN(Y_LC13,Y_LOC4);
                INCR(CTR13); MVDLTA;BRANCH(QUIT);
          MV135:PUTDOWN(Y_LC13,Y_LOC5);
                INCR(CTR13); MVDLTA;BRANCH(QUIT);
          MV136:PUTDOWN(Y_LC13,Y_LOC6);
                INCR(CTR13); MVDLTA;BRANCH(QUIT);
    CART_14:    TESTC(CTR14, 6, EXCEP14); TESTC(CTR14, 5, MV146);
                TESTC(CTR14, 4, MV145); TESTC(CTR14, 3, MV144);
                TESTC(CTR14, 2, MV143); TESTC(CTR14, 1, MV142);
                TESTC(CTR14, 0, MV141);
          MV141:PUTDOWN(Y_LC14,Y_LOC1);
                INCR(CTR14); MVDLTA;BRANCH(QUIT);
          MV142:PUTDOWN(Y_LC14,Y_LOC2);
                INCR(CTR14); MVDLTA;BRANCH(QUIT);
          MV143:PUTDOWN(Y_LC14,Y_LOC3);
                INCR(CTR14); MVDLTA;BRANCH(QUIT);
          MV144:PUTDOWN(Y_LC14,Y_LOC4);
                INCR(CTR14); MVDLTA;BRANCH(QUIT);
          MV145:PUTDOWN(Y_LC14,Y_LOC5);
                INCR(CTR14); MVDLTA;BRANCH(QUIT);
          MV146:PUTDOWN(Y_LC14,Y_LOC6);
                INCR(CTR14); MVDLTA;BRANCH(QUIT);
    EXCEP14:    SETC(CTR14,0);    BREAKPOINT; BRANCH(MAG_2);
    MAG_2:      SETC(MAG_NO,2);
TESTC(CTR1,6,N1);BRANCH(N2);N1:SETC(CTR1,0);
N2:TESTC(CTR2,6,N3);BRANCH(N4);N3:SETC(CTR2,0);
N4:TESTC(CTR3,6,N5);BRANCH(N6);N5:SETC(CTR3,0);
N6:TESTC(CTR4,6,N7);BRANCH(N8);N7:SETC(CTR4,0);
N8:TESTC(CTR5,6,N9);BRANCH(N10);N9:SETC(CTR5,0);
N10:TESTC(CTR6,6,N11);BRANCH(N12);N11:SETC(CTR6,0);
N12:TESTC(CTR7,6,N13);BRANCH(N14);N13:SETC(CTR7,0);
N14:TESTC(CTR8,6,N15);BRANCH(N16);N15:SETC(CTR8,0);
N16:TESTC(CTR9,6,N17);BRANCH(N18);N17:SETC(CTR9,0);
```

```
N18:TESTC(CTR10,6,N19);BRANCH(N20);N19:SETC(CTR10,0);
N20:TESTC(CTR11,6,N21);BRANCH(N22);N21:SETC(CTR11,0);
N22:TESTC(CTR12,6,N23);BRANCH(N24);N23:SETC(CTR12,0);
N24:TESTC(CTR13,6,N25);BRANCH(N26);N25:SETC(CTR13,0);
N26:TESTC(CTR14,6,N27);BRANCH(N28);N27:SETC(CTR14,0);
                  --MAGAZINE OVER. LIGHTS ON FOR MAGAZINE REPLENISHMENT
N28:  QUIT:
      END;

 MAGZ2 : SUBR(DESTN);                                --BRANCH TO LABEL

Y_LC1  : NEW   880;   Y_LC2 : NEW 847;
Y_LC3  : NEW   814;   Y_LC4 : NEW 781;
Y_LC5  : NEW   748;   Y_LC6 : NEW 715;
Y_LC7  : NEW   682;   Y_LC8 : NEW 649;
Y_LC9  : NEW   616;   Y_LC10 : NEW 583;
Y_LC11 : NEW   550;   Y_LC12 : NEW 517;
Y_LC13 : NEW   484;   Y_LC14 : NEW 451;

X_LC1 : NEW    -280;   X_LC2 : NEW -315;
X_LC3 : NEW    -350;   X_LC4 : NEW -385;
X_LC5 : NEW    -420;   X_LC6 : NEW -455;

SETC(C1,1);  SETC(C2,2);  SETC(C3,3);
  SETC(C4,4);  SETC(C5,5);  SETC(C6,6);
  SETC(C7,7);  SETC(C8,8);  SETC(C9,9);
  SETC(C10,10);  SETC(C11,11);  SETC(C12,12);
  SETC(C13,13);  SETC(C14,14);

TESTC(C1,DESTN, CART_1);      TESTC(C2, DESTN, CART_2);
TESTC(C3,DESTN, CART_3);      TESTC(C4, DESTN, CART_4);
TESTC(C5,DESTN, CART_5);      TESTC(C6, DESTN, CART_6);
TESTC(C7,DESTN, CART_7);      TESTC(C8, DESTN, CART_8);
TESTC(C9,DESTN, CART_9);      TESTC(C10, DESTN, CART_10);
TESTC(C11,DESTN, CART_11);    TESTC(C12, DESTN, CART_12);
TESTC(C13,DESTN, CART_13);    TESTC(C14, DESTN, CART_14);
      CART_1:
                  TESTC(CAT1, 6, EXCEP1); TESTC(CAT1, 5, MV16);
                  TESTC(CAT1, 4, MV15);   TESTC(CAT1, 3, MV14);
                  TESTC(CAT1, 2, MV13);   TESTC(CAT1, 1, MV12);
                  TESTC(CAT1, 0, MV11);
            MV11: PUTDOWN2(Y_LC1,X_LC1);
                  INCR(CAT1); MVDLTA2; BRANCH(QUIT);
            MV12: PUTDOWN2(Y_LC1,X_LC2);
                  INCR(CAT1); MVDLTA2; BRANCH(QUIT);
            MV13: PUTDOWN2(Y_LC1,X_LC3);
                  INCR(CAT1); MVDLTA2; BRANCH(QUIT);
            MV14: PUTDOWN2(Y_LC1,X_LC4);
                  INCR(CAT1); MVDLTA2; BRANCH(QUIT);
            MV15: PUTDOWN2(Y_LC1,X_LC5);
                  INCR(CAT1); MVDLTA2; BRANCH(QUIT);
            MV16: PUTDOWN2(Y_LC1,X_LC6);
                  INCR(CAT1); MVDLTA2;BRANCH(QUIT);
            EXCEP1:                        --MOVE TO NEXT MAGAZINE
                  SETC(CAT1,0);
```

```
                BREAKPOINT;
                BRANCH(MA_1);
CART_2:         TESTC(CAT2, 6, EXCEP2); TESTC(CAT2, 5, MV26);
                TESTC(CAT2, 4, MV25);   TESTC(CAT2, 3, MV24);
                TESTC(CAT2, 2, MV23);   TESTC(CAT2, 1, MV22);
                TESTC(CAT2, 0, MV21);
        MV21:   PUTDOWN2(Y_LC2, X_LC1);
                INCR(CAT2); MVDLTA2;BRANCH(QUIT);
        MV22:   PUTDOWN2(Y_LC2,X_LC2);
                INCR(CAT2); MVDLTA2;BRANCH(QUIT);
        MV23:   PUTDOWN2(Y_LC2,X_LC3);
                INCR(CAT2); MVDLTA2;BRANCH(QUIT);
        MV24:   PUTDOWN2(Y_LC2, X_LC4);
                INCR(CAT2); MVDLTA2;BRANCH(QUIT);
        MV25:   PUTDOWN2(Y_LC2, X_LC5);
                INCR(CAT2); MVDLTA2;BRANCH(QUIT);
        MV26:   PUTDOWN2(Y_LC2,X_LC6);
                INCR(CAT2); MVDLTA2;BRANCH(QUIT);
EXCEP2:         SETC(CAT2,0);
                BREAKPOINT;
                BRANCH(MA_1);
CART_3:         TESTC(CAT3, 6, EXCEP3); TESTC(CAT3, 5, MV36);
                TESTC(CAT3, 4, MV35);   TESTC(CAT3, 3, MV34);
                TESTC(CAT3, 2, MV33);   TESTC(CAT3, 1, MV32);
                TESTC(CAT3, 0, MV31);
        MV31:   PUTDOWN2(Y_LC3,X_LC1);
                INCR(CAT3); MVDLTA2;BRANCH(QUIT);
        MV32:   PUTDOWN2(Y_LC3,X_LC2);
                INCR(CAT3); MVDLTA2;BRANCH(QUIT);
        MV33:   PUTDOWN2(Y_LC3,X_LC3);
                INCR(CAT3); MVDLTA2;BRANCH(QUIT);
        MV34:   PUTDOWN2(Y_LC3,X_LC4);
                INCR(CAT3); MVDLTA2;BRANCH(QUIT);
        MV35:   PUTDOWN2(Y_LC3,X_LC5);
                INCR(CAT3); MVDLTA2;BRANCH(QUIT);
        MV36:   PUTDOWN2(Y_LC3, X_LC6);
                INCR(CAT3); MVDLTA2;BRANCH(QUIT);
EXCEP3:         SETC(CAT3,0);
                BREAKPOINT;
                BRANCH(MA_1);
CART_4:         TESTC(CAT4, 6, EXCEP4); TESTC(CAT4, 5, MV46);
                TESTC(CAT4, 4, MV45);   TESTC(CAT4, 3, MV44);
                TESTC(CAT4, 2, MV43);   TESTC(CAT4, 1, MV42);
                TESTC(CAT4, 0, MV41);
        MV41:   PUTDOWN2(Y_LC4,X_LC1);
                INCR(CAT4); MVDLTA2;BRANCH(QUIT);
        MV42:   PUTDOWN2(Y_LC4,X_LC2);
                INCR(CAT4); MVDLTA2;BRANCH(QUIT);
        MV43:   PUTDOWN2(Y_LC4,X_LC3);
                INCR(CAT4); MVDLTA2;BRANCH(QUIT);
        MV44:   PUTDOWN2(Y_LC4,X_LC4);
                INCR(CAT4); MVDLTA2;BRANCH(QUIT);
        MV45:   PUTDOWN2(Y_LC4,X_LC5);
                INCR(CAT4); MVDLTA2;BRANCH(QUIT);
        MV46:   PUTDOWN2(Y_LC4,X_LC6);
```

```
            INCR(CAT4); MVDLTA2;BRANCH(QUIT);
EXCEP4:     SETC(CAT4,0);
            BREAKPOINT;
            BRANCH(MA_1);
CART_5:    TESTC(CAT5, 6, EXCEP5); TESTC(CAT5, 5, MV56);
            TESTC(CAT5, 4, MV55); TESTC(CAT5, 3, MV54);
            TESTC(CAT5, 2, MV53); TESTC(CAT5, 1, MV52);
            TESTC(CAT5, 0, MV51);
      MV51:PUTDOWN2(Y_LC5,X_LC1);
            INCR(CAT5); MVDLTA2;BRANCH(QUIT);
      MV52:PUTDOWN2(Y_LC5,X_LC2);
            INCR(CAT5); MVDLTA2;BRANCH(QUIT);
      MV53:PUTDOWN2(Y_LC5,X_LC3);
            INCR(CAT5); MVDLTA2;BRANCH(QUIT);
      MV54:PUTDOWN2(Y_LC5,X_LC4);
            INCR(CAT5); MVDLTA2;BRANCH(QUIT);
      MV55:PUTDOWN2(Y_LC5,X_LC5);
            INCR(CAT5); MVDLTA2;BRANCH(QUIT);
      MV56:PUTDOWN2(Y_LC5,X_LC6);
            INCR(CAT5); MVDLTA2;BRANCH(QUIT);
EXCEP5:    SETC(CAT5,0);BRANCH(MA_1);
CART_6:     TESTC(CAT6, 6, EXCEP6); TESTC(CAT6, 5, MV66);
            TESTC(CAT6, 4, MV65);   TESTC(CAT6, 3, MV64);
            TESTC(CAT6, 2, MV63);   TESTC(CAT6, 1, MV62);
            TESTC(CAT6, 0, MV61);
      MV61: PUTDOWN2(Y_LC6,X_LC1);
            INCR(CAT6); MVDLTA2;BRANCH(QUIT);
      MV62: PUTDOWN2(Y_LC6,X_LC2);
            INCR(CAT6); MVDLTA2;BRANCH(QUIT);
      MV63: PUTDOWN2(Y_LC6,X_LC3);
            INCR(CAT6); MVDLTA2;BRANCH(QUIT);
      MV64: PUTDOWN2(Y_LC6,X_LC4);
            INCR(CAT6); MVDLTA2;BRANCH(QUIT);
      MV65: PUTDOWN2(Y_LC6,X_LC5);
            INCR(CAT6); MVDLTA2;BRANCH(QUIT);
      MV66: PUTDOWN2(Y_LC6,X_LC6);
            INCR(CAT6); MVDLTA2;BRANCH(QUIT);
EXCEP6:     TESTC(CAT14,6,NEXT6);          --TEST IF CART 13
            BRANCH(CART_14);               --IS FULL
NEXT6:      SETC(CAT6,0);                  --MOVE TO NEXT
            SETC(CAT14,0);                 --MAGAZINE
            BRANCH(MA_1);
CART_7:    TESTC(CAT7, 6, EXCEP7); TESTC(CAT7, 5, MV76);
            TESTC(CAT7, 4, MV75); TESTC(CAT7, 3, MV74);
            TESTC(CAT7, 2, MV73); TESTC(CAT7, 1, MV72);
            TESTC(CAT7, 0, MV71);
      MV71:PUTDOWN2(Y_LC7,X_LC1);
            INCR(CAT7); MVDLTA2;BRANCH(QUIT);
      MV72:PUTDOWN2(Y_LC7,X_LC2);
            INCR(CAT7); MVDLTA2;BRANCH(QUIT);
      MV73:PUTDOWN2(Y_LC7,X_LC3);
            INCR(CAT7); MVDLTA2;BRANCH(QUIT);
      MV74:PUTDOWN2(Y_LC7,X_LC4);
            INCR(CAT7); MVDLTA2;BRANCH(QUIT);
      MV75:PUTDOWN2(Y_LC7,X_LC5);
```

```
            INCR(CAT7); MVDLTA2;BRANCH(QUIT);
      MV76:PUTDOWN2(Y_LC7,X_LC6);
            INCR(CAT7); MVDLTA2;BRANCH(QUIT);
EXCEP7:     TESTC(CAT7,6,NEXT7);
            BRANCH(CART_7);
NEXT7:      TESTC(CAT13,6,NEXT13);
            BRANCH(CART_13);
NEXT13:     SETC(CAT7,0);
            SETC(CAT13,0);
            BRANCH(MA_1);
CART_8:     TESTC(CAT8, 6, EXCEP8); TESTC(CAT8, 5, MV88);
            TESTC(CAT8, 4, MV85);   TESTC(CAT8, 3, MV84);
            TESTC(CAT8, 2, MV83);   TESTC(CAT8, 1, MV82);
            TESTC(CAT8, 0, MV81);
      MV81: PUTDOWN2(Y_LC8,X_LC1);
            INCR(CAT8); MVDLTA2;BRANCH(QUIT);
      MV82: PUTDOWN2(Y_LC8,X_LC2);
            INCR(CAT8); MVDLTA2;BRANCH(QUIT);
      MV83: PUTDOWN2(Y_LC8,X_LC3);
            INCR(CAT8); MVDLTA2;BRANCH(QUIT);
      MV84: PUTDOWN2(Y_LC8,X_LC4);
            INCR(CAT8); MVDLTA2;BRANCH(QUIT);
      MV85: PUTDOWN2(Y_LC8,X_LC5);
            INCR(CAT8); MVDLTA2;BRANCH(QUIT);
      MV88: PUTDOWN2(Y_LC8,X_LC6);
            INCR(CAT8); MVDLTA2;BRANCH(QUIT);
EXCEP8:     TESTC(CAT8,6,NEXT88);
            BRANCH(CART_8);
NEXT88:     TESTC(CAT9,6,NEXT89);
            BRANCH(CART_9);
NEXT89:     TESTC(CAT10,6,NEXT810);
            BRANCH(CART_10);
NEXT810:    TESTC(CAT11,6,NEXT811);
            BRANCH(CART_11);
NEXT811:    TESTC(CAT12,6,NEXTMAG);
            BRANCH(CART_12);
NEXTMAG:    SETC(CAT8,0); SETC(CAT9,0); SETC(CAT10,0);
            SETC(CAT11,0); SETC(CAT12,0);
            BRANCH(MA_1);
CART_9:     TESTC(CAT9, 6, EXCEP8); TESTC(CAT9, 5, MV96);
            TESTC(CAT9, 4, MV95);   TESTC(CAT9, 3, MV94);
            TESTC(CAT9, 2, MV93);   TESTC(CAT9, 1, MV92);
            TESTC(CAT9, 0, MV91);
      MV91: PUTDOWN2(Y_LC9,X_LC1);
            INCR(CAT9); MVDLTA2;BRANCH(QUIT);
      MV92: PUTDOWN2(Y_LC9,X_LC2);
            INCR(CAT9); MVDLTA2;BRANCH(QUIT);
      MV93: PUTDOWN2(Y_LC9,X_LC3);
            INCR(CAT9); MVDLTA2;BRANCH(QUIT);
      MV94: PUTDOWN2(Y_LC9,X_LC4);
            INCR(CAT9); MVDLTA2;BRANCH(QUIT);
      MV95: PUTDOWN2(Y_LC9,X_LC5);
            INCR(CAT9); MVDLTA2;BRANCH(QUIT);
      MV96: PUTDOWN2(Y_LC9,X_LC6);
            INCR(CAT9); MVDLTA2;BRANCH(QUIT);
```

```
CART_10:     TESTC(CAT10, 6, EXCEP8); TESTC(CAT10, 5, MV106);
            TESTC(CAT10, 4, MV105);   TESTC(CAT10, 3, MV104);
            TESTC(CAT10, 2, MV103);   TESTC(CAT10, 1, MV102);
            TESTC(CAT10, 0, MV101);
      MV101: PUTDOWN2(Y_LC10,X_LC1);
            INCR(CAT10); MVDLTA2;BRANCH(QUIT);
      MV102: PUTDOWN2(Y_LC10,X_LC2);
            INCR(CAT10); MVDLTA2;BRANCH(QUIT);
      MV103: PUTDOWN2(Y_LC10,X_LC3);
            INCR(CAT10); MVDLTA2;BRANCH(QUIT);
      MV104: PUTDOWN2(Y_LC10,X_LC4);
            INCR(CAT10); MVDLTA2;BRANCH(QUIT);
      MV105: PUTDOWN2(Y_LC10,X_LC5);
            INCR(CAT10); MVDLTA2;BRANCH(QUIT);
      MV106: PUTDOWN2(Y_LC10,X_LC6);
            INCR(CAT10); MVDLTA2;BRANCH(QUIT);
CART_11:     TESTC(CAT11, 6, EXCEP8); TESTC(CAT11, 5, MV116);
            TESTC(CAT11, 4, MV115);   TESTC(CAT11, 3, MV114);
            TESTC(CAT11, 2, MV113);   TESTC(CAT11, 1, MV112);
            TESTC(CAT11, 0, MV111);
      MV111: PUTDOWN2(Y_LC11,X_LC1);
            INCR(CAT11); MVDLTA2;BRANCH(QUIT);
      MV112: PUTDOWN2(Y_LC11,X_LC2);
            INCR(CAT11); MVDLTA2;BRANCH(QUIT);
      MV113: PUTDOWN2(Y_LC11,X_LC3);
            INCR(CAT11); MVDLTA2;BRANCH(QUIT);
      MV114: PUTDOWN2(Y_LC11,X_LC4);
            INCR(CAT11); MVDLTA2;BRANCH(QUIT);
      MV115: PUTDOWN2(Y_LC11,X_LC5);
            INCR(CAT11); MVDLTA2;BRANCH(QUIT);
      MV116: PUTDOWN2(Y_LC11,X_LC6);
            INCR(CAT11); MVDLTA2;BRANCH(QUIT);
CART_12:    TESTC(CAT12, 6, EXCEP8);TESTC(CAT12, 5, MV126);
            TESTC(CAT12, 4, MV125); TESTC(CAT12, 3, MV124);
            TESTC(CAT12, 2, MV123); TESTC(CAT12, 1, MV122);
            TESTC(CAT12, 0, MV121);
      MV121:PUTDOWN2(Y_LC12,X_LC1);
            INCR(CAT12); MVDLTA2;BRANCH(QUIT);
      MV122:PUTDOWN2(Y_LC12,X_LC2);
            INCR(CAT12); MVDLTA2;BRANCH(QUIT);
      MV123:PUTDOWN2(Y_LC12,X_LC3);
            INCR(CAT12); MVDLTA2;BRANCH(QUIT);
      MV124:PUTDOWN2(Y_LC12,X_LC4);
            INCR(CAT12); MVDLTA2;BRANCH(QUIT);
      MV125:PUTDOWN2(Y_LC12,X_LC5);
            INCR(CAT12); MVDLTA2;BRANCH(QUIT);
      MV126:PUTDOWN2(Y_LC12,X_LC6);
            INCR(CAT12); MVDLTA2;BRANCH(QUIT);
CART_13:    TESTC(CAT13, 6, EXCEP7); TESTC(CAT13, 5, MV136);
            TESTC(CAT13, 4, MV135); TESTC(CAT13, 3, MV134);
            TESTC(CAT13, 2, MV133); TESTC(CAT13, 1, MV132);
            TESTC(CAT13, 0, MV131);
      MV131:PUTDOWN2(Y_LC13,X_LC1);
            INCR(CAT13); MVDLTA2;BRANCH(QUIT);
      MV132:PUTDOWN2(Y_LC13,X_LC2);
```

```
                INCR(CAT13); MVDLTA2;BRANCH(QUIT);
          MV133:PUTDOWN2(Y_LC13,X_LC3);
                INCR(CAT13); MVDLTA2;BRANCH(QUIT);
          MV134:PUTDOWN2(Y_LC13,X_LC4);
                INCR(CAT13); MVDLTA2;BRANCH(QUIT);
          MV135:PUTDOWN2(Y_LC13,X_LC5);
                INCR(CAT13); MVDLTA2;BRANCH(QUIT);
          MV136:PUTDOWN2(Y_LC13,X_LC6);
                INCR(CAT13); MVDLTA2;BRANCH(QUIT);
                BRANCH(CART_11);
    CART_14:    TESTC(CAT14, 6, EXCEP6); TESTC(CAT14, 5, MV146);
                TESTC(CAT14, 4, MV145); TESTC(CAT14, 3, MV144);
                TESTC(CAT14, 2, MV143); TESTC(CAT14, 1, MV142);
                TESTC(CAT14, 0, MV141);
          MV141:PUTDOWN2(Y_LC14,X_LC1);
                INCR(CAT14); MVDLTA2;BRANCH(QUIT);
          MV142:PUTDOWN2(Y_LC14,X_LC2);
                INCR(CAT14); MVDLTA2;BRANCH(QUIT);
          MV143:PUTDOWN2(Y_LC14,X_LC3);
                INCR(CAT14); MVDLTA2;BRANCH(QUIT);
          MV144:PUTDOWN2(Y_LC14,X_LC4);
                INCR(CAT14); MVDLTA2;BRANCH(QUIT);
          MV145:PUTDOWN2(Y_LC14,X_LC5);
                INCR(CAT14); MVDLTA2;BRANCH(QUIT);
          MV146:PUTDOWN2(Y_LC14,X_LC6);
                INCR(CAT14); MVDLTA2;BRANCH(QUIT);
    MA_1:      SETC(MAG_NO,0);
TESTC(CAT1,6,N1);BRANCH(N2);N1:SETC(CAT1,0);
N2:TESTC(CAT2,6,N3);BRANCH(N4);N3:SETC(CAT2,0);
N4:TESTC(CAT3,6,N5);BRANCH(N6);N5:SETC(CAT3,0);
N6:TESTC(CAT4,6,N7);BRANCH(N8);N7:SETC(CAT4,0);
N8:TESTC(CAT5,6,N9);BRANCH(N10);N9:SETC(CAT5,0);
N10:TESTC(CAT6,6,N11);BRANCH(N12);N11:SETC(CAT6,0);
N12:TESTC(CAT7,6,N13);BRANCH(N14);N13:SETC(CAT7,0);
N14:TESTC(CAT8,6,N15);BRANCH(N16);N15:SETC(CAT8,0);
N16:TESTC(CAT9,6,N17);BRANCH(N18);N17:SETC(CAT9,0);
N18:TESTC(CAT10,6,N19);BRANCH(N20);N19:SETC(CAT10,0);
N20:TESTC(CAT11,6,N21);BRANCH(N22);N21:SETC(CAT11,0);
N22:TESTC(CAT12,6,N23);BRANCH(N24);N23:SETC(CAT12,0);
N24:TESTC(CAT13,6,N25);BRANCH(N26);N25:SETC(CAT13,0);
N26:TESTC(CAT14,6,N27);BRANCH(N28);N27:SETC(CAT14,0);
                --MAGAZINE 2 OVER. LIGHTS ON FOR REPLENISHMENT
N28:  QUIT:
      END;

      COMBINE : SUBR(XOR,YOR,DEST1,DEST2);
            PICKUP(XOR,YOR);
            TESTC(MAG_NO,2,MTWO);
            TESTC(MAG_NO,0,MONE);
      MTWO:
            MAGZ2(DEST2);TESTC(MAG_NO,0,MONE);
            PUTPLATE(XOR,YOR);
            BRANCH(QUIT);
      MONE:
            MAGZ1(DEST1);TESTC(MAG_NO,2,MTWO);
```

```
            PUTPLATE(XOR,YOR);
            BRANCH(QUIT);
      QUIT:
      END;

      ITERATE('COMBINE',X,Y,DES,DES2);
      PMOVE(PT(1030,0,0,0));
      BREAKPOINT;
      --PLATE OVER, LIGHTS ON FOR NEW PLATE
END;
```

Table A16: Theta 1 Payload Data

Theta 1 (rads)	Average Time (secs) 0 lb	8 lb	16 lb	Emulator Prediction (secs)
0.01	33.5	30.6	29.6	33.58
0.02	45.2	45.7	45.3	45.05
0.79	141.2	140.2	140.7	140.33
1.05	164.9	164.6	164.5	165.17
1.57	211.5	211.7	211.8	211.79
2.09	258.9	258.5	258.6	258.72
2.36	282.9	282.8	282.3	283.05
3.14	353.9	353.4	353.4	354.03
3.49	384.0	384.5	384.6	384.14

*Tests consisted of repeating the angular move 100 times (50 cycles).

Table A17: Theta 2 Payload Data

Theta 2 (rads)	Average Time (secs) 0 lb	8 lb	16 lb	Emulator Prediction (secs)
0.01	32.2	32.7	33.2	31.79
0.07	54.9	54.9	55.3	54.56
0.79	113.2	113.2	113.2	113.31
1.05	128.6	129.1	128.9	128.82
1.57	159.2	159.6	159.3	159.29
2.09	189.9	190.0	190.1	189.68
2.79	230.8	230.9	231.0	230.72

*Tests consisted of repeating the angular move 100 times (50 cycles).

Table A18: Z-Axis Payload Data

Z-Axis (mm)	Average Time (secs) 0 lb	8 lb	16 lb	Emulator Prediction (secs)
- 10	10.48	10.63	10.53	8.73
- 80	19.36	19.44	19.42	19.36
-180	27.74	27.92	28.11	27.74
-250	34.20	33.80	33.73	34.20

*Test consisted of 9.5 cycles.

A METHOD FOR PRECISE ROBOT CYCLE TIME ESTIMATION

by

LAURA L. CRANMER

B.S., Kansas State University, 1988

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering
KANSAS STATE UNIVERSITY
Manhattan, Kansas
1989

ABSTRACT

This thesis presents a method developed to estimate robot cycle time without knowledge of the control system characteristics. The investigation into robot cycle time estimation was the result of a joint research effort by IBM and KSU. To complete the IBM-KSU research project a method was needed to estimate (within ±5%) cycle time of an IBM 7547 SCARA Robot. The method had to be quick and simple to meet the project time requirements. A literature search was conducted and no established accurate method was found. The empirical methods presented in the literature provided varying accuracy levels. The inaccuracy was primarily due to an inadequate characterization of the robot control system. Precision modeling of robot arms is difficult without detailed information on the control system characteristics. Such information, however, is not released by the manufacturers, so the problem remains how to characterize the robot experimentally and develop precise cycle time estimates.

For the IBM-KSU research project an IBM 7547 SCARA Robot was characterized empirically in a precise manner, and an emulator was developed to verify the characterization. This provided cycle time estimates within ±1%, comparing favorably to estimates developed for an IBM 7535 based on a known control system. From this process a

generalized method was developed to estimate robot cycle times precisely in a quick and easy fashion.